109

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 17/50, 13/10, 13/12, 9/455 — A1

(11) International Publication Number: **WO 00/41101**

(43) International Publication Date: 13 July 2000 (13.07.00)

(21) International Application Number: PCT/US00/00292

(22) International Filing Date: 5 January 2000 (05.01.00)

(30) Priority Data:

09/228,542	6 January 1999 (06.01.99)	US
09/336,445	18 June 1999 (18.06.99)	US
09/336,284	18 June 1999 (18.06.99)	US
09/443,175	19 November 1999 (19.11.99)	US

(71) Applicant *(for all designated States except US)*: SIMUTECH CORPORATION [US/US]; Five Lincoln Center, Suite 360, 10200 SW Greenburg Road, Portland, OR 97223 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: EVANS, Edward, N. [US/US]; 2929 Monogram Avenue, Long Beach, CA 90815 (US). JURASEK, David [US/US]; 40607 NW Monarch Lane, Banks, OR 97106 (US). HUGGINS, Jay, V. [US/US]; 2023 SE Market, Portland, OR 97214 (US).

(74) Agent: SIEGEL, Timothy, E.; Suite 200, 5125 SW Macadam Avenue, Portland, OR 97201 (US).

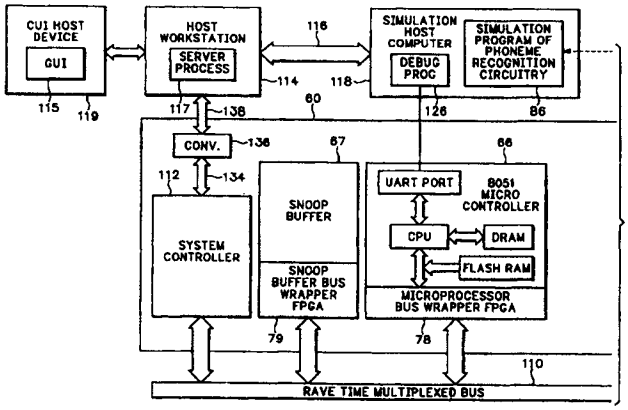
(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published

With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: APPARATUS AND METHOD FOR VERIFYING A MULTI-COMPONENT ELECTRONIC DESIGN



(57) Abstract

A verification engine (60) for verifying the design of a target system (10) having a plurality of components interconnected by a plurality of target system buses is disclosed. The verification engine (60) comprises a first hardware model (70) and a second hardware model (72), both configured as a component and having a set of hardware model input/output pins (147, 168). In addition, a first bus wrapper (82) is connected to the first hardware model (70) and a second bus wrapper (84) is connected to the second hardware model (72). Further, a set of bus lines (110) are each connected to the first bus wrapper (82) and the second bus wrapper (84). A system controller (112) is connected to at least some of the bus lines (110) and is adapted to transmit a sequence of time synchronization information to each bus wrapper control block by way of the bus lines (110). Finally, responsive to a predetermined one of the time slot numbers (3) both of the control blocks switch at least one input/output pin into communicative contact with the bus line (110) so that at least one input/output line from the first hardware model (70) is connected to an input/output line of the second hardware model (72).

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

APPARATUS AND METHOD FOR VERIFYING A MULTI-COMPONENT ELECTRONIC DESIGN

This Patent Cooperation Treaty Application Claims Priority from the following four United States Patent Applications: Serial No. 09/228,542, filed January 6, 1999, Applications Serial No. 09/336,445 and 09/336,284, both filed on June 18, 1999, and Serial No. 09/443,175, filed on November 19, 1999.

BACKGROUND OF THE INVENTION

The present invention is related to an apparatus and method for performing developmental testing on a target system electronic design that includes many multi-transistor components. Such a design could be implemented as a system-on-a-chip or on a PC board.

As IC design complexity increases so does the time required to verify each design. A first step in typical current design verification methodology is to divide a design into various functional blocks, and then to design and verify each block separately. These blocks (also referred to as "components") may be from 50 gates to 100,000 gates or more in complexity and may require computer simulation runs of between a few hours and a few days to verify the block to a first order of confidence. In the context of this application the term "component" refers to this type of block.

A great challenge is presented, however, by the necessity of verifying the performance of an entire target system that is composed of a group of these already verified blocks. Because a target system design may include several million gates, a week of computer time may be needed to simulate the entire design. Moreover, a new simulation run

must be performed each time the design is changed, greatly slowing the design process. In addition, a target system simulation can only be executed when a complete circuit description in electronic file format (a "net list") is available. In the future, it will be increasingly typical for a foundry to manufacture target systems that include some components that are proprietary to the foundry and other components that are designed by the target system designer. Typically, no net lists will be available for the foundry proprietary components.

Further complicating the process, most electronic systems today are "embedded systems" in the respect that they include both hardware and embedded software components. In the past, the dividing line was relatively simple. A microprocessor was chosen as the core of the system and this processor was then interfaced to its surrounding environment with application specific integrated circuits (ASICs) and other custom logic. This completed the basic hardware system, and a prototype board was built and used for software development. For tasks that were speed critical, some software routines could be implemented in custom hardware, but with a significant cost both in production and in development. Today, IC designers have the ability to manufacture large ICs that implement many tasks in hardware that formerly were performed by software, thus creating much faster systems. Because much of the hardware in a system like this is designed to work with specific software, this requires that both the hardware and software be developed together. Unfortunately, software verification requires an order of magnitude more simulation patterns to verify than does hardware verification. There is currently no means of running these verification tests until a hardware prototype of the system exists, typically at the completion of

hardware design. If a hardware error is uncovered during the software testing, it forces a difficult decision between a very expensive change to the finalized hardware design or a cumbersome, and perhaps slow, software work-around.

To ease the task of debugging software that resides in an embedded system, various special software tools have been developed. These packages typically include a ROM monitor component that resides in a read only memory assembly that is accessible to the microprocessor. When the microprocessor is booted it begins operation using the instructions in the ROM monitor. Another debug package component resides on a test computer that is connected to some of the pins of the microprocessor. The test computer can instruct the ROM monitor to load the software program that runs on the microcontroller with a breakpoint that causes operation to jump to the ROM monitor. The ROM monitor instructions cause the microcontroller CPU to send specified register contents out through the pins that are connected to the test computer for display to a developer. At present it is generally not possible to use this type of debug package to its fullest effect until all the hardware components are completed and an entire system is ready to be tested. Alternatively, the debug package could be used without the hardware components. This will, of course, not find problems that occur in the interaction of the software and the hardware. The early use of such a debug package would be tremendously beneficial to software developers in their efforts to debug software prior to the time when an entire system has been constructed.

In order to speed up the verification time of a target system design, various methods have been used. These generally fall into three categories: hardware modelers, emulators and simulation accelerators.

Hardware modelers address the above noted problematic situation in which a net list does not exist for one of the target system blocks. In this situation it is generally the case that a physical embodiment of the block exists in the form of an IC or a bonded out IC core (a portion of an IC that has been extracted and equipped with connector pins). A hardware modeler is designed to connect such a physical embodiment to a computer executing a simulation model (a "simulator"). Unfortunately, a single hardware modeler only connects a single physical embodiment to the simulator. Although an additional physical embodiment could be connected to the simulator by an additional hardware modeler, communications between the two physical embodiments would have to be implemented by the simulator, greatly slowing system performance. Because of this, hardware modelers generally do not greatly accelerate the simulation process for complex systems.

At least one current hardware modeler allows a user to place one or more physical components on adapter boards that are then inserted into the modeler. This modeler is also connected to a simulator. The modeler allows the designer to incorporate the components on the adapter boards into an event-driven simulation, thus obtaining an accurate model of the component without the need for a net list. Unfortunately, all connections between the physical embodiments must, nevertheless, go through the simulator. If a microprocessor is placed in the hardware modeler, it could be used to do hardware/software co-verification, except that because all communications must be routed through the simulator it is far too slow to be useful for that purpose.

One recently released product is targeted at taking a microprocessor IC or bonded out core and connecting

it to an event-driven simulator, utilizing software debugging tools to allow hardware and software designers to use a real hardware model of the core processor during system design verification. The overall speed of execution of a system being designed with this product, however, will always be limited by the speed of the event-driven simulator, where the major portion of the design exists. This will be too slow for effective hardware/software simulation.

Another available product is intended to be a system level rapid prototyping solution. The product consists of a generic prototyping board that has two prototype areas with prepunched holes and no ICs attached. The prototype areas are where the customer places ICs or field programmable gate arrays (FPGAs) that represent different building blocks in an IC or printed circuit board design. All of the pins in both prototyping areas can then be routed (any pin to any pin) via a set of proprietary custom crossbar switches. These crossbar switches are programmable, so that mistakes in routing can easily be corrected. This product facilitates the creation of a flexible prototype that can run close to a target system's design speed and can be used for software development after the hardware design is complete or for system level verification. Unfortunately, because there is no integration of either software debugging tools or an event-driven simulator with this product, it does not allow easy hardware/software co-verification during the development stages. Moreover, when the system includes three or more components this device is not of great utility.

Simulation accelerators are basically customized parallel processing computers that speed up the simulation run time significantly. Accelerators, however, do not

address the problem noted above with respect to any system component for which no net list is available.

Emulators use FPGAs to emulate the design being created. Using software tools, the design netlist is subdivided between a hardware emulation set of FPGAs. An interconnect set of FPGAs is used to reconfigurably interconnect the hardware emulation set of FPGAs. Emulators are significantly faster than other simulation methods, but they make significant restrictions on the format of the design source files and are difficult to expand significantly to emulate a system on a chip.

The new target system design methodology will rely heavily upon the reuse of virtual components (VCs). The VCs would typically be available to the developer in the form of FPGA loadable encoded files. In addition a contract foundry would have access to a set of production tools, including photolithography masks, for etching the VC onto silicon. A new business area is being developed that consists of designing and selling various VCs to be incorporated into target systems. The actual system design will normally be a synchronous design using a collection of VCs and normally adding several new custom design blocks to create a complete system. The partitioning of the design into medium sized design blocks will be done by the designer and these blocks will be interconnected by two to four standard buses.

Those familiar with electronic design will readily recognize that for a complex design that is to be implemented on an integrated circuit, the creation of a schematic diagram set is only the first step in a lengthy process. The physical layout of the integrated circuit must then be determined. This is a sometimes-difficult process, especially given that etching into silicon imposes several restrictions on the physical layout. Once a design has been

rendered into an integrated circuit (IC), the information and production tools, including photolithography masks that enable a foundry to produce the IC is a valuable piece of intellectual property that is typically kept as a trade secret. The desire to keep the product design as a trade secret conflicts, however, with the desire to sell the use of the design or a component part of the design to interested parties, who may not be directly competing with the design originator. For example, a first company may have designed a signal-processing component (perhaps including thousands of transistors) that it may use on a chip that performs signal processing for a radar system. A second company may wish to use the same component as part of a speech recognition IC, or perhaps as part of an IC that would be used for analyzing seismic signals.

To address this problem, the foundry that produces the IC for the first company may be licensed to produce the component as part of an overall design for a second company. In this manner, the design may be licensed without the second company ever having access to the details of the design. The second company, however, is at this point faced with a troublesome design question: How can it verify the design of a system that uses a component the internal workings of which are unknown to the second company? To ease the situation of the second company the first company may supply an FPGA loadable file that models the component but is encoded so that the second company cannot access any circuit description. Alternatively an IC or bonded-out IC core may be used. This, however, nevertheless leaves system integration as a difficult task, requiring the breadboarding of an FPGA configured to model the component into the rest of the system. Moreover, even at the test stage, without knowing the internal workings of a component, the task of

interconnecting the chip to other components may be problematic.

An additional problem that is encountered in testing electronic systems is that ensuring some level of completeness of test. To completely test every state of a complex system is impractical, as it would take decades if not centuries. A more achievable level of test completeness is to have toggled (driven to both binary states) every component pin. There is currently no hardware system available that easily permits a user to test a target system to at least this level of completeness.

In a system in which at least a portion of the circuit is modeled in hardware a problem is encountered when a behavioral model in a simulator represents a portion of the system being verified. A designer may choose to do this when he has not designed the internal logic of a component but has developed a behavioral model, that is, a computer program that mimics the input-to-output relationships that the designer anticipates will be present in the finished component. It may be possible to model every other component with a hardware model resident in the verification engine, but heretofore it would not have been possible to host a behavioral model in the verification engine.

This limitation poses at least two troubling problems. A first problem is encountered when a portion of the design or a portion of the target system itself is not finalized to the point where a net list is available. A related problem is encountered when the target system controls or otherwise interfaces with an electro-mechanical device, which cannot be emulated by a field programmable gate array (FPGA).

Physical world interfaces also present a problem to circuit verifiers. For example, the electronics in a

nuclear power plant must be thoroughly verified prior to being used operationally. It is difficult to verify this control circuitry, however, without using it to operate a nuclear power plant. One method of verifying the operation of the control circuitry without risking a nuclear mishap is to include a behavioral model of the electro-mechanical portions of the nuclear power plant in a simulation program that simulates the control circuitry. If the control circuitry is being modeled in hardware, there is currently no ready facility for modeling the behavior of an associated electro-mechanical element that is not communicatively limited as described above.

SUMMARY OF INVENTION

In a first preferred aspect, the present invention is a verification engine for verifying the design of a target system having a plurality of components interconnected by a plurality of target system buses. The verification engine comprises a first hardware model and a second hardware model, both configured as a said component and having a set of hardware model input/output pins. In addition, a first bus wrapper is connected to the first hardware model and a second bus wrapper is connected to the second hardware model. Further, a set of bus lines are each connected to the first bus wrapper and the second bus wrapper. Each bus wrapper also has switchable communicative circuitry that switchably communicatively connects each hardware model input/output pin to a bus line and has a control block controlling the switchable communicative circuitry. A system controller that is connected to at least some of the bus lines is adapted to transmit a sequence of time synchronization information to each said bus wrapper control block by way of the bus lines. The time

synchronization information is sufficient to permit said control blocks to uniformly determine a time slot number. In response, the control blocks uniformly determine the time slot number and in response thereto each control block switches at least one input/output pin into communicative contact with a said bus line so that at least one input/output line from the first hardware model is connected to an input/output line of the second hardware model.

In a separate preferred aspect, the present invention is a verification engine for verifying the design of a target system having a plurality of components interconnected by a plurality of target system buses. The verification engine comprises a plurality of reconfigurable bus wrappers, each having a bus wrapper/hardware model set of pins and a bus wrapper/bus line set of pins, switchable communicative circuitry that switchably communicatively connects each said bus wrapper/hardware model pin to a bus wrapper/bus line pin and a control block controlling said switchable communicative circuitry. A set of bus lines each has a bus line/bus wrapper pin for each bus wrapper. Each bus line/bus wrapper pin is connected to a bus wrapper/bus line pin. A system controller is connected to at least some of the bus lines and is adapted to transmit time synchronization information sufficient for the control blocks to uniformly determine a time slot number. In addition, responsive to the time synchronization information, each control block determines the time slot number and responsive to a predetermined one of the time slot numbers, switches at least one bus wrapper/hardware model pin into communicative contact with a said bus wrapper/bus line pin.

In an additional separate preferred aspect, the present invention is a method of determining drive direction

between a first circuit node that is communicatively connected to a first driver input pin of a first driver and a second circuit node that is communicatively connected to a second driver input pin of a second driver. The first driver and the second driver are connected in mutual opposition by a connecting line. Moreover, the first and second circuit nodes are tentatively commanded to be connected by enabling either the first driver or the second driver with input from the first circuit node or the second circuit node respectively. The method comprises automatically forming a first test result by determining if the first circuit node is being driven and recording the first test result in a first format.

In still another additional separate preferred aspect, the present invention is a bus adapted to selectively couple a first electrical component having a set of first component input/output pins to a second electrical component having a set of second component input/output pins. The bus comprises a first bus wrapper having a set of first bus wrapper-to-component pins adapted to be connected to said first component input/output pins; a set of first bus wrapper bus line pins; a set of latches, each latch being switchably communicatively connected to a said first bus wrapper-to-component pin and to a said first bus wrapper bus line pin; and a local condition sensor. A second bus wrapper has a set of second bus wrapper-to-component pins adapted to be connected to the second component input/output pins; a set of second bus wrapper bus line pins; and a set of latches, each latch being switchably communicatively connected to a said second bus wrapper-to-component pin and to a said second bus wrapper bus line pin. A set of bus line conductors connect each first bus wrapper bus line pin to a second bus wrapper bus line pin. A system controller

has a set of system controller bus line pins connected to the set of bus line conductors and adapted to transmit to said bus wrapper a sequence of time slot numbers defining a sequence of time slots. Each bus wrapper is adapted to determine, upon receipt of each time slot number enumerator which, if any, of its latches is active during the time slot number and in the first bus wrapper is further adapted to determine, at least in part by said local condition sensor and said time slot number, which active latches will be switched into a transmitting communicative coupling with a first bus wrapper-to-component pin and which will be switched into a transmitting communicative coupling with a first bus wrapper bus line pin.

In still another additional separate preferred aspect, the present invention is a method for verification testing of a target system design made up of a plurality of components connected by a plurality of buses that permit a defined flow of data between said components, the method comprising connecting hardware models of at least some of the components with a time multiplexed bus and controlling the time multiplexed bus to permit said defined flow of data between the hardware models.

In a further separate preferred aspect, the present invention is either an apparatus, or a virtual apparatus comprising a computer readable memory device bearing a net list for an apparatus, for interfacing to an integrated circuit that contains bi-directional pins. The apparatus or virtual apparatus comprises a soft drive adapted to soft drive the bi-directional pin high during a high drive time and to soft drive the bi-directional pin low during a low drive time. In addition, a first flip-flop is configured to save the output of the bi-directional pin during its high drive time and a second flip-flop is

configured to save the output of the bi-directional pin during its low drive time. In addition, an Exclusive OR gate has a first input that is connected to the output of the first flip-flop and a second input that is connected to the output of the second flip-flop. The output of the Exclusive OR gate, after the high and low drive times, is thereby indicative of pin drive direction. A computer readable media bearing a circuit description for a circuit according to the description of this paragraph is yet another separate preferred aspect of the present invention.

In a still further preferred aspect the present invention comprises a device for facilitating the prototyping of a target system electronic design by a developer. The target system electronic design that is to be prototyped should include a first component, having an internal construction which may be unknown to the developer and having a first pin, a second component and a third component that transmits a first signal to the first pin of the first component. The device comprises a computer including a socket and a memory assembly bearing a program written in a simulation language and adapted to simulate the third component and to form a first model state that models the first signal. A computer software object, stored in the memory of the computer is adapted to translate the first model state into a first socket command thereby causing a first socket output signal, which models the first signal, to be transmitted from the socket. Further, a communicative assembly is operatively coupled to the computer and a first hardware model that is adapted to model the first component and includes a first model pin that models the first pin and is operatively connected to the communicative assembly. In addition, a second hardware model is adapted to model the second component and is operatively connected to the

communicative assembly. Finally, a server assembly is interconnected between the socket and the communicative assembly and is adapted to create the first communicative assembly signal, which models the first signal, in response to the first socket output signal and wherein the communicative assembly is adapted to pass the first communicative assembly signal to the first model pin.

In a still further separate preferred aspect the present invention is a device for facilitating the prototyping of a target system electronic design by a developer. The target system electronic design should include a first component, having an internal construction which may be unknown to the developer and having a first pin, a second component and a third component that receives a first signal from the first pin of the first component and forms a first state in response to the first signal. The device itself comprises a communicative assembly and a first hardware model adapted to model the first component and including a first model pin that models the first pin. Moreover, the first model pin is operatively connected to the communicative assembly and the first hardware model is further adapted to transmit the first signal from the first pin. The device further includes a second hardware model adapted to model the second component and that is operatively connected to the communicative assembly. A computer includes a socket and a memory assembly that bears a program that simulates the first component and receives a first socket signal over the socket. The computer, further, is operatively connected to the communicative assembly. A computer software object is stored in the memory of the computer and is adapted to model the third component and to receive the first socket signal and to form a first model state, which models the first state, in response to the

first socket signal. A server assembly is interconnected between the socket and the communicative assembly is adapted to create and transmit the first socket signal, which models the first signal, in response to receiving the first communicative assembly signal.

In a still further separate preferred aspect, the present invention is a device for facilitating the prototyping of a target system electronic design by a developer. The electronic design should include a microprocessor having registers bearing register contents, a computer memory assembly bearing a program adapted to run on the microprocessor and a component having an internal construction that may be unknown to the developer and a first communicative linking assembly connecting the microprocessor to the computer memory and the component. The device comprises a hardware implemented microprocessor model adapted to model the microprocessor at least in part by forming model register contents. A component model adapted to model the component model memory assembly bears the program and memory contents. A model communicative assembly connects the model memory assembly and the component model to the microprocessor model. A computer and a linking and reporting assembly links the computer to the model communicative assembly and adapted to deliver selected ones of the memory contents to the computer at developer specified points in the program.

In a still further separate preferred aspect the present invention is a verification engine for testing a set of components, each of which has a set of input/output pins. The verification engine has the ability to ensure toggle coverage for a set of component pins. The verification engine comprises an interconnective assembly having a toggle detector for each of the input/output pins. The output of

each toggle detector is recorded in a memory space. A detecting and reporting assembly detects and reports to a user which, if any, of the input/output pins has not been toggled.

In a still further separate preferred aspect, the present invention is a method of mapping a system level hardware description, containing a set of target system components, each of which is designated by a target system component name in the system level hardware description, into a set of virtual components, each of which is present in a library of hardware models and is designated by a virtual component name. The method comprises the steps of parsing the system level hardware description into the set of target system components and displaying a target system component name representative of a target system component. The virtual component name is displayed and the user is permitted to match the target system component name to the virtual component name thereby designating hardware model corresponding to the virtual component name as a model for target system component.

In a still further separate preferred aspect, the present invention is a method of partitioning a system level hardware description, containing a set of target system components, into a first subset of components that are to be modeled in a verification engine and a second subset of components that are to be modeled in a simulator. Each target system component is designated by a target system component name. The method comprises the steps of parsing the system level hardware description into the set of target system components, displaying a target system component name and permitting a user to designate the target system component corresponding to the target system component name as belonging to the first subset of components or the second

set of components.

In a still further separate preferred aspect, the present invention is a method, for use by a designer, to efficiently design a first portion of an integrated circuit (IC) having a component for which internal structure and IC production information exists but is unavailable to the designer and an interconnection scheme for interconnecting the component to a second portion of the IC for which a second portion model exists. The method comprises the steps of providing a reconfigurable interconnect assembly, an FPGA and a set of virtual component designators, each of which may be implemented by an available FPGA loadable file and for which internal structure IC production information exists but is unavailable to the designer. The designer uses an electronic design software package to create a higher-level system description of the first portion, the higher-level system description including a first functional description of the component. The designer then accesses the set of virtual components and selects a virtual component from the library based on the first functional description. A first FPGA loadable file is available to implement the virtual component on an FPGA. The first FPGA loadable file is loaded onto the first FPGA. The reconfigurable interconnect assembly is used to interconnect the first FPGA to the second portion model to form a multicomponent electronic system. The multicomponent electronic system is tested to verify its operation and is corrected when an error is discovered to form a corrected multicomponent electronic system.

In still another additional separate preferred aspect, the present invention is a verification engine adapted to model a target system having a set of components. The verification engine comprises a multiplicity of

hardware models, each hardware model representing a component. A bus links together the hardware models and includes a system controller. A computer runs a simulation program of a component that is not represented by a hardware model and is thereby a simulated component. In addition, the computer is operatively connected to the system controller. A set of latches is operatively connected to the bus. The system controller directs all output signals from the simulation program to the set of latches and the set of latches represents the simulated component to the bus and the hardware models.

The foregoing and other objectives, features, and advantages of the invention will be more readily understood upon consideration of the following detailed description of the invention, taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a fictitious target system presented for purposes of illustrating the preferred method and apparatus of the present invention.

FIG. 2A is one half of a block diagram of a verification engine configured to model the target system of FIG. 1 according to the preferred method and upon the preferred apparatus of the present invention.

FIG. 2B combines with FIG. 2A to form a complete block of a verification engine configured to model the target system of FIG. 1 according to the preferred method and upon the preferred apparatus of the present invention.

FIG. 3 is a table of the verification engine time multiplexed bus time period assignment of the verification engine configuration of FIG. 2.

FIG. 4 is a physical side view of the preferred apparatus embodiment of the present invention.

FIG. 4A is a physical side view of an alternative preferred apparatus embodiment of the present invention.

FIG. 4B is a physical side view of an additional alternative preferred apparatus embodiment of the present invention.

FIG. 4C is a physical side view of a further alternative preferred apparatus embodiment of the present invention.

FIG. 5 is a functional block diagram of a core card of FIG. 4 according to the present invention.

FIG. 6 is a flow diagram of the preferred overall test method of the present invention.

FIG. 7 is a schematic diagram of a drive direction detection circuit according to the preferred method of the present invention.

FIG. 8 is a schematic diagram of a bit of a fight/float detection logic circuit.

FIG. 9 is a block diagram of the active portions of the system controller of the apparatus of FIG. 4 during regular mode

FIG. 10 is a block diagram of the active portions of the system controller of the apparatus of FIG. 4 during debug mode

FIG. 11 is a flow chart of the TMB fight/float-checking algorithm according to the preferred embodiment of the present invention.

FIG. 12 is a block diagram of a fictitious portion of an integrated circuit (IC) design.

FIG. 13 is a block diagram of the fictitious portion of the integrated circuit design of FIG. 12, as implemented in a virtual prototyping system according to the

present invention.

FIG. 14 is a graphical user interface screen display for instance selection, shown before instance selection, that forms a part of the present invention.

FIG. 15 is a graphical user interface screen display for instance selection, shown after instance selection, that forms a part of the present invention.

FIG. 16 is a graphical user interface screen display for VC selection that forms a part of the present invention.

FIG. 17 is a graphical user interface screen display for input/output pin selection that forms a part of the present invention.

FIG. 18 is a graphical user interface screen display for clock domain selection that forms a part of the present invention.

FIG. 19 is a graphical user interface screen display for reset sequence selection that forms a part of the present invention.

FIG. 20 is a block diagram of alternative method of implementing the connection between virtual components in a virtual prototyping system according to the present invention.

FIG. 21 is a block diagram of a fictitious motor and motor controller assembly that is presented for purposes of illustrating the present invention.

FIG. 22 is a block diagram of a verification engine configured to verify the operation of the motor and motor controller assembly of FIG. 1, according to the present invention.

FIG. 23 is a block diagram of a fictitious signal router that is presented for purposes of illustrating the present invention.

FIG. 24 is a block diagram of a verification engine configured to verify the operation of the signal router of FIG. 23, according to the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

I. A Fictitious Target System

FIG. 1 shows a fictitious voice recognition target system for the purposes of illustrating how a target system may be verified by a preferred embodiment of the present invention. In the following description the term "target system" is used as a qualification to indicate an action or part of the target system design, as opposed to being a part of the verification engine. For example, a target system clock signal is a clock signal that is part of the target system design that is being verified. A verification engine system clock signal is a signal that the verification engine uses to coordinate its own timing.

Target system 10 includes an analog-to-digital (A/D) converter 12, a discrete Fourier transform (DFT) engine 14, a phoneme recognition processor 16, a data processor 18 (an 8051 microcontroller), and a DRAM 20, containing a program for the data processor 18 to permit word recognition based on the phoneme information from processor 16. Additionally, context recognition post processor 40 examines the words produced by data processor 18 to determine, in the case of homonyms, which homonym is correct, or in the case in which a word is pronounced in a partially unintelligible manner, what it is that the speaker meant to say. The words corrected by the context recognition post processor 40 are returned to the data processor 18 so that data processor 18 can adapt to the speech pattern of the speaker. Data processor 18 then

determines the phonemes from the word received from the context recognition processor 40 and sends these phonemes back to phoneme recognition processor 16 so that it to may adapt to the speaker's speech pattern.

Proposed target system 10 is connected to the outside world by an input line 22, which is typically connected to microphone or a tape player. An output bus 23 delivers a series of letters in ASCII format, representing the recognized words, to some receiving device, such as a display system. Although a target system clock 36 generally drives the system 10 at 200 MHZ, the output bus is clocked at 1/1024 of this speed. Because the output bus is longer and uses more energy than the internal buses of target system 10, the slower rate is desirable to save energy, and is still fast enough to deliver the output of target system 10. To implement the slower bus speed, the post processor 40 includes two clock inputs, "A" and "B," where "A" drives the context recognition circuitry and "B" clocks out the output of the post processor 40 and is driven by a divide by 1024 counter 42.

The internal connections of target system 10 include a 16 bit A/D output data bus 24, which transports the digitized sound to DFT engine 14, a 96 bit DFT output data bus 26, which transports the DFT output data to the phoneme recognition processor 16. A 40 bit bidirectional multi-port data processor bus 28, transmits the recognized phonemes together with commentary data to data processor 18; transmits the occasional corrected phoneme from data processor 18 to phoneme recognition processor 16; transmits words to context recognition post processor 40; transmits corrected words from post processor 40 to data processor 18; and permits data processor 18 to obtain instructions from DRAM 20. Additionally, a first set of discrete data lines

32 link the data processor 18 to the phoneme recognition circuitry 16 as a second set of discrete data lines 34 link the data processor 16 to the A/D converter 12.

In system operation, as noted above, A/D samples input line 22 every 65 microseconds. The target system clock operates at a 200 MHZ rate. Because the human ear cannot detect transitions that take place in less than 10 milliseconds, however, a spectrum is delivered over the 96 bit DFT bus only 100 times per second. When a spectrum is delivered, however, it requires 1,000 clock cycles, or 5 microseconds to deliver to phoneme recognition circuitry 16. Phoneme recognition circuitry 16 sends a phoneme code to data processor 18 every time it recognizes a phoneme. This event occurs aperiodically and results in an interrupt being sent to data processor 18, which then controls data processing bus 28 to retrieve the phoneme code and an identifying tag. Phoneme recognition happens on average about 9 times per second. Word recognition, performed by data processor 18 occurs about 3 times per second on average. A context recognition clarification occurs on average about every 4 seconds and must be fed backwards to the data processor, which in turn, sends the correct phoneme, identified by the identifying tag, back to the phoneme recognition circuit.

II. The Modeling of the Fictitious Target System

FIG. 2 shows a verification engine 60 according to the present invention. All of the components of target system 10, except for the phoneme recognition circuitry 16 are modeled by a hardware model, with the data processor 18 being modeled by an FPGA 66 configured to mimic the operation of an actual 8051 microcontroller. It is also possible to use a bonded out IC core, which is a portion of

an IC design that is extracted and equipped with input/output pins. In many situations, the microprocessor may be sold as a chip, which may be used directly. DFT engine 14 and A/D converter 12 are simulated by a DFT engine IC 64 and an A/D converter IC 62, respectively. A central memory unit 68 that includes a Small-Outline Dual-Inline Memory Module (SODIMM) module models the DRAM 20. The context recognition post processor 40 is modeled by a field programmable gate array (FPGA) 70 that has been configured appropriately with an FPGA loadable file. A dummy hardware model FPGA 72 stores the input/output data for simulated phoneme recognition circuitry 16. This linking mechanism will be discussed in greater detail below. Each one of hardware models 62, 64, 66, 68, 70 and 72 is linked to a verification engine 128 bit time multiplexed bus (TMB) 110 by way of a reconfigurable, appropriately configured "bus wrapper" FPGA 74, 76, 78, 80, 82 and 84 respectively. A system controller 112 controls time multiplexed bus 110. Additionally, a "snoop" buffer 67 is provided with a "snoop" buffer bus wrapper 79, to permit a user to view the messages sent on TMB 110 after an interruption of regular mode operation. These values are useful for debugging. A tape player 89 accommodating a tape recording of human speech used as an input to A/D converter IC 62, thereby providing a set of test vectors to verification engine 60.

Referring to FIG. 3, TMB 110 is caused to sequentially model the operation of the buses connecting the components of the target system by separating its operation into a sequence of time slot numbers, and performing at least a portion of the communications handled by at least one target system bus in each time slot number. The sequence of time slot numbers is continuously iterated during regular mode operation of verification engine 60.

Specifically, bus 110 performs the function of the A/D output bus 24 and DFT bus 26 (time slot number 1), data processor bus 28 (time slot number 2), discrete signals (time slot number 3), and output bus 23 (time slot number 4). In an alternative preferred embodiment no time slot number is sent from the system controller 112, but each bus wrapper 144 independently but uniformly computes the time slot number from the system clock and/or other signals sent from the system controller 112.

III. The Verification Engine Connection to Outside Tools

Referring to FIG. 2, the system controller 112 is connected to a standard Intel® 960 bus 134. The Intel® 960 bus is connected by way of an adapter card 136 to a standard Peripheral Connections Interface (PCI) bus 138, which is, in turn, connected to workstation 114, which controls system controller 112.

Host workstation 114 also hosts a server process 117, which is described below. Additionally connected to the host workstation 114 is a graphical user interface (GUI) host computer 119, which hosts a GUI 115 that facilitates a developer's configuration and use of the verification engine 60. Further connected to host workstation 114 by way of a standard local area network 116 is a simulation host computer 118 that is executing a simulation program 86 of phoneme recognition circuitry 16.

A debug program 126 is also hosted by computer 118 and interacts with a cooperative program in the flash ram of the 8051 microcontroller for the purpose of delivering parameter values to the developer seated at computer 118

The GUI 115 receives system design information from the developer. The developer then uses the GUI to divide the target system design between a portion that is to

be simulated by a simulation program 86 and that which resides within the verification engine 60. The GUI 115 then creates appropriate design data to be inserted within the simulation program 86, data conversion tables for a server process 117 (discussed below and also typically resident on work station 114) and configuration data to be loaded into the verification engine 60. In addition, the GUI 115 facilitates entry by the user of any FPGA configuration files that are to be loaded into FPGAs contained within the verification engine 60 and representing a portion of the user design. When verification system problems occur, the GUI can also be used to control the verification engine in debug mode. Among other operation, the GUI 115 can facilitate data retrieval from snoop buffer 67. The GUI 115 can then format and display this information to the developer.

The server process 117 running on the host work station 114 is responsible for the real time continuous communication between the various software programs, such as simulation program 86, interacting with the verification engine 60 and the GUI 115. When a request comes from a host computer 118 or 119, the server 117 reformats that request into an appropriate format for the engine system controller 112 and transmits it to the controller 112 via the connecting PCI bus 138. When the verification engine 60 has data to be transferred to a host computer 118 or 119, it first transfers this data to the server 117, which reformats it and then returns the data to the appropriate host computer 118 or 119.

The simulation program 86 communicates with the verification engine implementation of the rest of the target system 10 by way of a dummy hardware module 72. Dummy hardware module 72 includes a set of latches that causes the

exterior conductors of hardware model 72 to mimic the behavior of the input/output pins of phoneme recognition circuitry 16 in target system 10. Bus wrapper 84 contains the standard registers that all the bus wrappers 74-84 include (discussed later). Accordingly, dummy hardware model 72 interfaces to the rest of the verification engine 60 in the same way that all of the other hardware models 62-70 do so that during regular verification engine 60 operation dummy hardware model 72 appears to the rest of verification engine 60, like an actual phoneme recognition circuit 16.

As skilled persons will recognize, dummy hardware model 72 must be updated by simulation program 86 each time simulation program 86 has new output. Likewise, new data from the verification engine implementation other portions of target system 10 must be sent to simulation program 86 as well as to dummy hardware model 72. This communication is performed in a series of steps that begins with the dummy hardware model 72 receiving new, changed input from the TMB 110 and notifying the system controller 112 of this event by way of an interrupt. System controller 112 then fetches the data words stored in dummy hardware model 72 and relays them to the simulation host computer 118.

The system controller 112 freezes the operation of the TMB until it receives a response from the simulation host computer 118, which it relays to dummy hardware model 72. In this manner dummy hardware model 72 presents to the TMB 110 the output states that would be presented by the phoneme recognition circuitry 16 in the actual target system 10. Although the time necessary to make these updates does slow system operation, it is a fairly small time demand compared with the much longer time necessary for simulation

program 86 to execute and to send data to and receive data from the host work station 114.

Dummy hardware model 72 also includes an external output pin 73 and an external input pin 75, principally for the purpose of issuing or receiving an interrupt signal during testing.

One potential problem with verification engine 60 that connects to a simulator occurs with the DFT bus 26, if the phoneme recognition circuitry is simulated in software. This bus is actually delivering data only .05% of the time the target system is in operation. But the bus is technically always in operation, simply clocking out a set of all zeroes when no actual data transmission is occurring. Because phoneme recognition is simulated (see FIG. 2) on a computer 118 that is connected by a network 116 to the host workstation 114 the length of time needed to transmit data to it is far longer than it would be for a circuit that was hardware modeled. If the dummy, all zero, output had to be sent to computer 118 every clock cycle, system operation would slowed to a crawl. To avoid this situation dummy hardware module 72 is configured to send an interrupt to system controller 112 only when its input from TMB 110 changes. As a result, as long as DFT IC 64 is not emitting a DFT, operation of verification engine 60 proceeds at a rapid pace, relatively close to actual anticipated operational speed. As noted above, a DFT is only emitted every 10 milliseconds. Although verification engine operation slows down greatly during the 1,000 target system clock cycles required for delivery of a DFT, this represents only 2% of all total target system clock cycles.

One possible advantage of verification engine 60 is that system components in various stages of design may be linked together for system test. In this example, the data

processor 18, DFT engine 14 and A/D converter 12 are all completed designs. As of this writing it is becoming increasingly easy to buy such designs for incorporation into a larger design. In this illustration, however, the phoneme recognition circuitry 16 and the context recognition post processor 40 are both still in development. In the test configuration shown, the design of the phoneme recognition circuitry 16 is being most carefully scrutinized and may be changed during test, in the simulator. It is possible that in a subsequent round of testing the design for the phoneme recognition circuitry 16 will be loaded into an FPGA and the design for the post processor 40 will be simulated and further debugged and developed.

FIG. 4 is a generalized physical drawing of verification engine 60, showing the hardware model/bus wrapper cards, referred to as core cards, 120, as plug in units to bus pin sockets 122 of time multiplex bus physical backplane 124. Each core card includes a bus wrapper 140 and a hardware model 142. FIG. 4A shows an alternative preferred embodiment in which each core card 120 is divided into a pair of cards: A bus wrapper card 120a and a mating hardware model card 120b. FIG. 4B shows an alternative preferred embodiment in which a single FPGA 141 serves as both a bus wrapper and a hardware model. In the alternative preferred embodiment shown in FIG. 4C the system controller 112 is implemented not as a separate, plug-in card but as a series of components on the bottom of the physical backplane 124.

IV. Bus Wrapper Configuration

Referring to FIG. 5 each generalized bus wrapper 140 must take the transient content of the TMB 110 bus data lines (BDLs) 144 and make this content available to the

inputs of the attached generalized hardware model 142 at the time slot numbers and at the appropriate ones of a set of hardware model data pins 147. (In this application the term "pin" is used even if only a set of wires connect a bus wrapper 140 with a hardware model 142. In such a case the portion of each wire closest to the hardware model would be considered a hardware model pin and likewise for the bus wrapper 140.) Similarly, each bus wrapper 140 must take the output of the attached hardware model pins 147 and make it available to the TMB BELs 144 during each appropriate time slot number. Both the input and the output functions are accomplished by a set of bidirectional drivers 146. In the target system 10 being modeled and verified, however, many components may communicate by using more than one bus. Therefore, some bus wrappers 140 must serve the function of more than one virtual bus wrapper for more than one virtual bus. In one preferred embodiment this feature is accomplished for up to three virtual buses by having three complete sets of registers 150, 152 and 154. Whether or not to use a particular register 150, 152 or 154 during a particular time slot is determined by a time slot number compare by control blocks 155, 156 and 157 respectively, using the input from a set of time slot bus lines 126, for each register set 150, 152 and 154. If a match is found, the associated registers become active during the current time slot. Primary register 1 (of set 150, 152 or 154), describes which pins 147 are active during the current time slot and these pins are placed into contact with the corresponding BELS 144. The manner in which they are placed into contact (i.e., whether they are driving or receiving) is described in detail in the discussion accompanying FIG. 7.

A two-bit op code is sent to each bus wrapper 140 on a pair of TMB op code lines 129. Included in the op code is a target system clock-indicating signal (see discussion of FIG. 9), which must be positive for any target system clock to be issued. In addition, for the clock input pins in a particular clock domain, there must be a match in the time slot number-compare register 130 corresponding to the clock domain. When this happens one of a set of target system pre-clock line drivers issues a clocking input to one of a set of target system pre-clock input pin(s) 168 in hardware model 142.

Referring to FIG. 3, for target system 10, time slot numbers 3 and 4 would be the sole entries in registers 130 with time slot 3 being associated with clock domain A, which includes every target system clock input pin except clock input pin B of post processor 40. As shown in FIG. 3 every time a time slot number 3 occurred the target system clock bit in the op code would be positive and every pin in clock domain A (see FIG. 3) would receive a clocking input. But, this bit would be positive only on every 1024th occurrence of time slot number 4, at which time (and only at this time) the sole pin of clock domain B would receive a clocking input.

In this preferred embodiment (shown in FIG. 5) each one of the registers is a full 128 bits wide, matching the number of bus data lines (BDLS) 144 on the TMB 110. In an alternative preferred embodiment there are six complete sets of registers, six-time slot decode matches and six sets target system pins to be connected but, due to space limitations, each register is only 64 bits wide.

Although in one preferred embodiment the bus wrappers 140 are implemented on FPGAs, in an alternative preferred embodiment, the bus wrappers are implemented as

hardwired circuits with a basic structure similar to that shown in FIG. 5. If the bus wrapper is slated to be used in conjunction with (and may share a core card with) an FPGA for implementing hardware models, the pin routing could be done in the hardware model FPGA, saving the bus wrapper from the need to do this task. If the bus wrapper is slated to be used in conjunction with a hardwired hardware model, such as an IC or a bonded out IC core, then it would also require a set of crossbar switches for pin routing from the IC pins 147 to TMB 110.

V. Test Verification Process

Referring to FIG. 6, the verification engine configuration shown in FIG. 2 may be configured and executed (after obtaining and installing the necessary ICs) by the steps shown, beginning with the identification of a component level system description in a higher level design language and the identification of FPGA loadable files (block 310) for those hardware items that will be modeled in FPGAs. These files will typically be available on computer readable media, such as a floppy disk or a CD-ROM. Next, the GUI program 115 attempts to match the pins listed in the FPGA loadable files with the pins of the higher-level target system design file, and the user is given an opportunity to correct the result (block 312).

In one preferred embodiment, the user commands the execution of a computer program in the verification engine host workstation 114 that designs the bus wrappers for each one of the hardware models (block 314). The output of this program is a set of net lists, which are translated into FPGA loadable files each of which is loaded into a bus wrapper FPGA (block 316), the hardware model FPGAs are also loaded at this time, together with the contents of central

memory 68 (the program for data processor 18 that is to be resident in DRAM 20 on target system 10).

In an alternative preferred embodiment, each bus wrapper net list is chosen from a predesigned set of net lists, for example, as described above, one design has three sets of register 150, 152 and 154 and another design has six sets of registers. Each bus wrapper is individualized to accommodate the different configurations of input, output and input-output pins, distributed among various target system buses, through the inclusion of a number of registers which are loaded by the host workstation after the FPGAs are loaded with the FPGA loadable files. These registers will be described in greater detail later on. At this point the verification engine is started (block 318), typically with a program stored in central memory 68. In addition a set of external stimulus data 319 may be introduced. For example tape player 89 feeds this data into A/D converter card 62, in the fictitious example verification set-up shown in FIG. 2.

At each stage of execution a problem may be detected (decision box 320). If no problem is detected execution continues. But if a problem is detected, the "snoop" buffer 67 may be read out to workstation 114 and reviewed to determine the cause of the problem (block 321). The snoop buffer 67 contains all of the BDL 144 values for the recent history of TMB 110. If this approach does not yield an answer (decision box 323), the debug (register) mode is used (block 322) to permit the human operator of workstation 114 to examine the contents of the registers in the core cards 120. If the problem can be ignored or corrected without redesigning the portion of the target system (decision box 324) being verified, then test may

continue (block 318). Otherwise, test is halted and redesign is initiated (block 326).

VI. Detecting Hardware Model Pin Drive Direction and Avoiding Bus Fight and Float Conditions

In testing circuitry that includes bidirectional pins a number of problems occur. First, there is the problem of determining which direction a bi-directional pin is being driven by the hardware model 142 at any particular time. This is unpredictable for a number of reasons. First, if the hardware model is an IC or a bonded out IC core, the manufacturer is likely to have retained an exclusive knowledge of some elements of the system design to keep an advantage over prospective imitators. Moreover, even if the full circuit description is available, as in the case of an FPGA that has been loaded with a circuit description, it entails considerable effort to predict the bi-directional pin states. The driving direction must be determined, however, for the bus wrapper to know whether to drive the pin 147 (using the bus output) or to allow the pin to drive the bus data line 144.

An additional problem is caused by the fact that the target system is not fully designed and verified. Because of this, there may be situations in which two pins attempt to drive the same bus connection (wire) during the same time slot. This is referred to as a "bus fight" and, because neither driving signal is received, as the sending circuit anticipates, it causes the system operation to go awry. In a related situation, neither one of a set of bidirectional pins that are connected during a particular time slot drives the connection, causing each pin to receive random noise rather than an anticipated signal. This condition is known as a "bus float." Again, this typically

causes system operation to go awry. In traditional testing this type of problem has been rather difficult to detect and a source of frustration to system verification personnel.

To avoid these problems, the verification engine 60 has a subsystem designed to determine for each bidirectional pin, whether it is driving or open for an input signal. In addition for each set of bidirectional pins that are to be connected during a particular time slot the system controller determines if there is a bus fight or a bus float.

TABLE 1

Secondary Register 8	Primary Register 1	Secondary Register 9	Drive Direction Option
0	0	x	1. TMB to target system Component
0	1	x	2. Target system to TMB Component
1	x	0	3. Drive Direction Determined by Local Logic and State of Target system Component Output Line States
1	x	1	4. Drive Direction Determined by Testing Pin

Referring to FIG. 7, switchable connective circuitry 146a is provided to buffer each hardware model target system pin or circuit node 144 without regard to whether the particular target system pin 144a is an input, output or bi-directional pin 144. The drive direction of each buffer is determined, in part by the core card registers from register set 150 as described in TABLE 1 and shown by the inputs to switches 182 and 184, which together determine how the pin will be treated. Drive direction options 1 and 2 are self-explanatory. Option 3, requires some specialized design for the bus wrapper. This would typically be implemented in cases where one particular pin acts as a signal that all or a portion of the pin in a hardware model have a particular direction for the present or future time slot. There are some types of target system internal buses in which the drive direction is signaled by a particular target system component pin. For this type of bus, local logic may be configured to test the state of the signal pin and base the drive direction of the other pins on the state of this pin.

In option 4, the drive direction of a pin is determined by testing the pin 147a to determine whether or not it is itself being driven. The circuitry that performs this function is a local condition sensor because it senses a condition that is encountered by the bus wrapper 140. This determination is made by finding if the pin 147a can be driven by a soft driver, that is, a high impedance driver. This soft driver is formed by a shift register 218, and resistor R1. The shift register 218 starts counting up from zero when a target system preclock A signal is received from block 130 (FIG. 5). On the first TMB clock cycle following the issuance of target system pre-clock A, an actual target

system clock is sent to the hardware model on target system clock A input line 168.

If an input driver 240 is enabled, pin 147a will accept input from driver 240 at the moment the target system clock signal is applied to line 168. If pin 147a is bidirectional, however, it may also form an output value shortly after the advent of the target system clock (i.e. before the next target system clock pulse). On the next system clock the shift register 218 disables input driver 240 by a low input to an AND gate 244. The purpose of this is to prevent any input to pin 147a during the period when it is being determined whether hardware model 142 is driving pin 147a as such input would interfere with the test and potentially conflict with output from pin 147a. Subsequently, shift register 218 emits a low and then a high on the line connected to resister R1.

If the hardware model pin 147a can be driven both high and low by this high impedance (due to resistor R1) output, then it is not being driven by the hardware model 142. In this event a first FLIP-FLOP 220 will have a different output than a second FLIP-FLOP 222 and the output of a drive direction XNOR gate 224 will be low, enabling the input driver 240, which sends the Q output of input FLIP-FLOP 225 to hardware model pin 147a. Otherwise, the output of XNOR gate 224 will be high, enabling an output driver 242, which sends the Q output of output FLIP-FLOP 226, which is from pin 147a, to TMB BDL 144a. The AND gates 260 and 262 report a fight or float condition respectively to fight and float registers 264 and 266, respectively. Such a fight or float condition could occur, for example, if primary register 1 had been selected to determine the drive direction, but the drive direction that this register had determined was in conflict with the test performed by shift

register 214 and FLIP-FLOPS 220 and 222. The contents of a pair of registers 264 and 266 are masked and ORED together to create an "error" output from the bus wrapper 140. The masking is performed because there may be instances in which a fight or a float is expected and/or acceptable. The mask is user configured to reflect this reality and prevent an error flag from being issued when an expected or acceptable fight or float has occurred.

Referring to FIGS. 8, 9 and 11 the task of detecting bus fight and bus float conditions at the system level is performed by both the system controller 112 and dummy hardware model 72. The system controller 112 (FIG. 9) has a fight mask application logic circuit 402 and a float mask application logic circuit 404 each composed of 128 bits, to prevent a fight or float error from being issued when a fight or float does not indicate an error. For example, without float mask application logic 404, when a particular TMB BDL 144 was not used, this would be erroneously reported as a fight error by the system controller. Dummy hardware model 72 merely records and then reports out the presence of either a "high," a "low," a "fight," or a "float" in a two bit format that is standard in simulation software programs. Because model 72 does not take action when an error is detected, there is no need to mask the fight and float detections to avoid declaring an error in an instance in which a fight or a float does not indicate an error condition.

In the system controller, a fight and float mask register memory 406 (FIG. 9) includes a 128 bit word to be loaded into the appropriate mask for every active time slot and each board that is active in the time slot. In the masks each "0" indicates a bit that should be checked for a fight or a float condition, respectively, and a "1" for each

bit that should not be checked in this manner. To facilitate the masking operation, a core card time slot register 408 indicates which core cards are active during each time slot. In "regular" mode operation, prior to each time slot, the pin drive register of each time slot active card is polled. That is, referring to FIG. 11, after the time slot N is incremented (block 620) the drive direction data (primary register 1 from register set 150, 152 or 154 depending on the time slot number comparison is read from each active core card in sequence (blocks 622 and 624 and decision boxes 626 and 628). Referring to FIG. 8, the results for each bit, line 147a (FIG. 8), are read into a fight/float detection circuit 410, comprising a first FLIP-FLOP 412, the output of which drives and two-input AND gate 414 the other input of which is, again, line 147a. The output of gate 414 is fed into a second FLIP-FLOP 416. A bus fight is indicated by the output of the FLIP-FLOP 416 going high, whereas a bus float is indicated by the output of the first FLIP-FLOP 412 remaining in a low state at the end of the process. The first FLIP-FLOP 412 outputs (stage 1) are masked with the float mask apply mechanism 402 for time slot N (block 632) and the second FLIP-FLOP 416 outputs (stage 2) are masked with the fight mask apply mechanism 404 (block 630). The test results are ORed together in block 558 and in block 570 and the resultant bit is ORed with the interrupts received from core cards 120 on at the system controller on core card interrupt pins 572. Because of this, when either a bus fight or a bus float is detected (decision boxes 634 and 636) for any of the pins an interrupt is sent to host workstation 114 (block 638). If no bus fight or bus float is detected the TMB 110 is activated to permit an actual of transfer between hardware models 142 (block 640).

To perform the polling described above it is necessary for system controller 112 to be able to address each core card 120 separately. There is, indeed, a mechanism for permitting this to be accomplished. There are six core card select lines 540 (see FIGS. 9 and 10) on the time multiplex bus. Each core card slot has a unique six-bit hardwired code, that when matched by the six card select lines 540 causes the selection of the inserted card. When the six lines represent the number 63, their highest possible number (in one particular embodiment), this indicates that no card is specifically selected and that all cards should respond to the time slot lines 126.

Two other submodes of regular mode operation, fast submode and check submode exist for system controller 112. In fast mode no polling is performed. A bus fight or float would go undetected, but the system would operate many times faster than in regular submode. Fast submode would typically be used when the target system had been largely verified and could be used with a fair degree of confidence. Check submode addresses the following problem.

Some systems that require verification may not be entirely synchronous. It is possible that at times an asynchronous signal will not be represented correctly in time due to the verification sequence of connections. For example, an asynchronous discrete signal might occur prior to an internal bus clock occurring in the actual system, but appear afterwards in verification engine 60. It is possible that one of the bus states would have been different had the asynchronous discrete signal been modeled in the correct time relationship to the bus communication. So, in check mode, to permit all asynchronous communications to have their proper effect each time slot number is repeated without any target system clock being issued until the

output of a first instance actuation is matched exactly by a second instance actuation.

VII. System Controller Operation

FIGS. 9 and 10 shows those portions of the system controller 112 that are active during regular mode and debug mode respectively. The input from the 960 bus 134 is coded so that a much wider array of information may be delivered to system controller 112 by workstation 114 than the limited number of wires on the 960 bus 134 would otherwise permit. Therefore, the ports shown on the left side of the system controller exist in address space rather than as physical pins.

The system controller 112 has three essential tasks. First it must load and test the core card FPGAs at the beginning of system test. Second, it must cycle TMB 110 in regular mode. This includes the fight/float checking described earlier. Third, in debug mode it must forward requests for information from the host workstation 114 to the core cards 120 and forward the informational responses from the core card 120 of interest back to the host workstation 114.

Referring to FIG. 10, the loading of FPGAs through an FPGA load port 508 is old in the art and will not be treated in great detail here. The information is received from the host workstation 114 on FPGA load port 506 and sent out to the core cards from FPGA load lines 508. A JTAG interface port 542 and a set of JTAG interface lines 544 permit standard FPGA testing to be conducted by host workstation 114 through system controller 112.

Referring to FIG. 9, in regular mode, a sequencer 510, beginning with the time slot number indicated by a beginning time slot number port 584 and with the board

indicated by beginning board No. port 584 sequences through each time slot, as indicated by the number of time slots port 513. For each time slot, to achieve the bus fight/float detection described above, through each core card (informed by register 408). (There are ports for loading mask storage 406 and register 408, but in the interests of conciseness they are not shown). The system clock 514, produced by frequency synthesizer 580, is the basic unit of time for verification engine 60. The frequency of the system clock 514, generated by a frequency synthesizer 590 may be set from workstation 114 through a port (not shown). This flexibility permits the clock to be run as fast as possible for a given set of system components while ensuring correct system operation. A cycle count port 512 sets the number of time slot cycles to be performed by the sequencer 510 before operation is automatically brought to a halt. In addition, several breakpoint conditions may be set on each core card 120 by appropriately loading the secondary register of the appropriate core card 120. The current time slot port 532 and current board number port 534 pass the quantities onto workstation 114, so that it can start operation where it was terminated by correctly setting the beginning board No. port 582 and beginning time slot No. port 584.

The OP CODE logic 516 issues an op code for each time slot over OP CODE lines 129. This code informs each core card of the following information:

I. Regular Mode (Regular or Debug mode chosen from Mode Select Port 536)

A. Generate or Don't generate a target system clock

B. Read/Write cycle (all selected cards both read & write data)

C. Read cycle: all selected cards read data to TMB

D. Write cycle: all selected cards write data to TMB

E. No operation

II. Debug Mode

A. Write or Read cycle

B. Primary Register Number (0-7)

Target system clock timing block 520 periodically commands a target system clock to be reflected in the OP CODE. This is treated by the bus wrappers as described in the discussion of FIG. 5. The periodicity of the commands for a target system clock is user defined as a periodic pattern for each time slot number. For example every other occurrence time slot number 7 may include a target system clock.

When a target system clock is issued every target system clock input pin that has been partitioned into the clock domain corresponding to the target system clock receives a clock input from its bus wrapper 140.

In regular mode, the input to a set of 128 bus data lines 144 is sent to a set of fight/float stage one and stage two latches 412 and 416. The fight and float masks are applied as described earlier by apply logic (AND gate banks) 402 and 404. A fight/float final detect block 558 ands together the results from the mask apply registers 402 and 404. This condition is ORED together by block 570 with the other core card interrupts from lines 572 and is made available to host workstation 114 on the interrupt port. But in debug mode the values on line set 144 are shunted directly to bus data port 556 (see FIG. 10), so that system controller 112 acts as a conduit from TMB 110 to host workstation 114.

FIG. 10 shows system controller 112 as it appears functionally during debug mode. The card selection feature and the registers in the core cards form the basis for debug mode, that permits the system controller 112 to assist a human operator of host workstation 114 to debug the system under test after a problem is detected. The primary registers may be addressed by primary register address bits received on port 552 and sent out as part of the OP CODE, as noted earlier, on OP CODE lines 129. The register contents are received over a set of bus data lines 144 and sent to workstation 114 from virtual bus port 556. Likewise, register contents may be received from workstation 114 on port 556 and sent to a core card that workstation 114 has chosen via the core card select port 576 (and sent over bus on core card select lines 540) on lines 144. By loading primary register #2 in this manner, any secondary register may be addressed. In addition, a particular time slot operation may be ordered by time slot select port 574.

VII VIRTUAL PROTOTYPING SYSTEM - DESCRIPTION OF FICTITIOUS DESIGN

Referring to FIG. 12, and for purposes of illustration, this application will describe a particular case in which a fictitious circuit designer wishes to design a portion 2010 of a fictitious prospective system-on-a-chip. The portion 2010 includes an 8051 microcontroller 2012, a digital signal processing engine 2014, a phoneme recognition circuit 2016 and a video processor 2020, which drives a liquid crystal display 2022. The designer uses a common electronics design tool that allows him to select and link together gates and higher-level components for his design with the results being displayed on a computer monitor. The designer selects a digital signal processing engine and a

video processor, each being represented by a circuit description of that is part of a library associated with the design tool, and places both elements onto his circuit diagram by a series of mouse clicks. The phoneme recognition circuit, however, must be built up from the gate level. The designer links the various elements together with a set of communicative lines, to which he gives names. As the designer labors to configure the phoneme recognition circuit 2016, he is able to use output from the DSP engine 2014(as provided to the design tool) to check its operation. The designer has decided to use an 8051 microcontroller 2012 and has decided to verify the software after the phoneme recognition circuit 2016 is completed, although he can write the software for the 8051 microcontroller 2012 sooner.

After the designer has completed his initial phoneme recognition circuit 2016 design, he wishes to verify the design of the four components working together with the 8051 microcontroller 2012 programmed according to an initial scheme. Of course, he needs a bank of digitized voice signals and he provides a floppy disk that bears these and that may be connected via a computer 2089 to the input of the DSP engine 2014. At this point, the designer has a circuit description of the DSP engine 2014, the phoneme recognition circuit 2016 and the video processor 2020 and has identified the microcontroller 2012 he wishes to use. All of these are linked together in a higher-level design language overall system description. He does not, however have a physical circuit layout for either the DSP processor 2014, the phoneme recognition circuit 2016 or the video processor 2020. At this point, the designer may start to use the virtual prototyping system 2050 that is the subject of this invention.

The goal of the designer is to configure a virtual prototyping system 2050 such as that shown in FIG. 13, which will be described in greater detail later. System 2050 includes a verification engine 2060 having a reconfigurable interconnect assembly 2110 that links together an 8051 microcontroller IC 2066, a DSP engine IC 2064, an FPGA configured as a video processor 2068 and a port 2112. A toggle verification and reporting system that includes toggle registers 2310 and reports out which pins, if any, have been left untoggled. A host workstation 2114, that hosts a server process 2117 connects verification engine 2060 to a simulation host computer 2118 hosting a simulation program 2086 (for simulating the operation of phoneme recognition circuitry 2016). An embedded software development computer 2800 hosts an embedded design environment 2810 that includes a debug tool 2812.

IX VIRTUAL PROTOTYPING SYSTEM - GRAPHIC USER INTERFACE

Also connected via workstation 2114 is a graphical user interface (GUI) host device 2119, hosting a GUI 2115. In one preferred embodiment the reconfigurable interconnect assembly 2110 is in the form shown and described in the parent and companion applications. A different preferred embodiment for assembly 2110 is shown in FIG. 20, which shows components 2064, 2066, 2068 and 2072 linked together by a set of bus wrappers 2140. In one preferred embodiment, each bus wrapper 2140 would include a set of bidirectional driving assemblies similar to that shown in FIG. 17of the companion application to accommodate bidirectional pins of the components. In the embodiment shown in FIG. 19 time multiplexing is unnecessary.

The GUI 2115 facilitates the configuration of virtual prototyping system 2050. To begin, the GUI 2115

prompts a user to indicate a file containing an overall system description, either in the Verilog language or in VHDL. After the user indicates this file, the GUI 2115 automatically parses the file in order to determine what components are represented in the system. Referring to FIG. 14, the GUI displays the target system components, the microcontroller 2012, phoneme recognition circuit 2016 and the DSP 2014, in column form to the user, who is then asked to designate (by the use of a mouse associated with workstation 2114) which components are to be modeled in the verification engine 2060. As shown in FIG. 15, the designer designates the DSP engine 2014 and the microcontroller 2012 and the video processor 2020. Because the phoneme recognition circuitry 2016 has just been designed and will in likelihood go through significant changes, the designer decides to simulate this part of the program, on the same workstation he has been using to design the system and which is connected to workstation 2114 by a network. The GUI presumes that an associated simulation-running computer 2118 will simulate the components indicated for the verification engine 2060.

Referring to FIG. 16, the GUI 2115 displays a column showing a library of virtual components (VCs) together with the column of components to be modeled in the verification engine. A VC may be in one of at least three basic forms. It may include a FPGA loadable file, or an IC or bonded out IC core. In any of these cases the VC package would include a pin description file. In the current example, a DSP engine IC and a 8051 microcontroller IC are already physically included in the verification engine 2060, and have associated pin description files in the GUI library. The designer is prompted to indicate a VC for each target system component modeled in the verification engine

2060. It is anticipated that packages of VCs will be sold to users and that users will also construct their own libraries of VCs. In some cases, a VC will be obtained and loaded into the workstation 2114 in anticipation of modeling a particular target system. In the present example, the designer finds a video processor VC that he is already familiar with and that has the same input/output scheme and operation characteristics as the video processor model he has been using in the design tool. Unlike that video processor model, however, the video processor VC he identifies in the GUI 2115 is associated with a complete set of production tools for making the video processor VC as part of a larger system-on-a-chip. Although this information, together with the actual Video processor netlist is unavailable to the designer, there is a foundry that has the production tools and will make an IC that includes the Video processor VC, for a fee. A coded, FPGA loadable file that is coupled with a pin list is, however, available and the designer in choosing the Video processor VC indicates that this associated FPGA loadable file will be used to model the Video processor engine component.

At this juncture the GUI 2115 matches the pin list for the Video processor VC with the system connective line names provided by the designer during the initial design process described earlier. Referring to FIG. 17, the results of this matching operation are displayed in side-by-side columns. In some cases, where a good match has been chosen, the GUI 2115 will be able to perform this matching operation with a high degree of confidence. At the opposite extreme, when the user makes an error and specifies a VC netlist file that does not at all match the system component that it is designated to model, most of the pin assignments will be left unmade. Between these two extremes it is

anticipated that there will be some cases in which the match is generally good, but in which there are a few pins that either the GUI 2115 cannot match or for which an uncertain match is formed. The level of certain of the match is color coded into the column display of matching pins with a first color indicating strong certainty, a second color indicating less certainty and another color indicating very little certainty or noting that no matching pin could be found. Although it may seem peculiar to the electrical engineering layperson that a virtual component could be designated that was a close but not exact match, many designs include some pins that are present as a convenience to the designer and/or perform some occasionally desirable function. Of course, if there were a disagreement between the major functional pins of the VC and a target system circuit description the VC would be unusable for verifying the target system circuit. But it is frequently possible to rectify a disagreement among the minor, convenience, pins by simply not using a VC pin or by a minor target system redesign to eliminate a pin that is only occasionally used, for convenience. The user is given the opportunity to change the pin mapping by pointing and clicking on the various column entries. This same process must be performed for the 8051-microcontroller 2066 connective pins and the video processor VC 2068 connective pins.

In the example case, the designer notes that two pins have been highlighted in the Video processor VC column, for which there is no equivalent in the target system circuit description. Fortunately, these are bus control pins, the use of which is optional. For the time being, the designer decides to leave these unconnected to the phoneme recognition circuitry, but notes their presence as something he may wish to exploit in the future.

In the next step, shown in FIG. 17, the GUI 2115 partitions the pins into separate clock domains. The user is prompted to correct any undesirable features of the clock domain partitioning performed by the GUI 2115 and to specify a clock rate for each domain. In the example case, all the clock input pins may share a common clock input signal.

Referring to FIG. 19, the user is next prompted to specify an initialization sequence for resetting the target system at the start of operation. This process is facilitated by a list of signals, any one of which may be selected and given an initialization input. In this case, the user specifies the manufacturer provided boot sequence of an 8051 chip.

Next, each discrete signal and target system bus is displayed to the user and the user is given the opportunity to specify bus signals that should be sent to the simulator (in addition to signals sent to the simulated portion of the target system, which are automatically sent to the simulation computer). The example system is so small that there is little opportunity to illustrate this feature. If the phoneme recognition circuit 2016 was not simulated, however, its output would be a natural item that a designer might wish to have forwarded to the simulation running computer for display.

Next the GUI 2115 must configure the verification engine reconfigurable interconnect assembly 2110. Regardless of the form of the interconnect assembly 2110, this will mean assigning component pins to connective assembly lines. For example, if the connective assembly takes the form of a single time multiplexed bus (as in the companion application), each active component pin must be assigned to a bus line and a time slot. For many target systems a very simple algorithm proceeding in an arbitrary

order may do this, much as a person might make the assignments. The result of this process is called the "configuration file" in which each target system signal is assigned to a bus line/time slot combination. The configuration file is used to design the bus wrappers. Frequently, however, existing bus wrappers may easily be used and no bus wrapper design is needed. Please see the companion patent application for a more complete description of the bus wrapper configuration and operation.

Finally, the GUI 2115 sends the configuration file to a server 2117, so that the server 2117 may arrange for communications between dummy software module 2086 and verification engine 2060. In order to understand how the server process does this, it is necessary to understand the structure of the of the simulation host computer 2118 to verification engine 2060 connection, which is described below.

X VIRTUAL PROTOTYPING SYSTEM - INTEGRATION OF VERIFICATION ENGINE AND SIMULATOR

Returning to FIG. 13, the phoneme recognition circuit 2016 is represented in the verification engine 2060 by a dummy hardware module 2072. As a sort of a mirror image to module 2072, a dummy software module 2710 represents, for the simulation program 2086, the portion of the target system 2010 components being verified in the verification engine 2060. Just as dummy hardware module 2072 is indistinguishable from a fully functional hardware module to the rest of the verification engine 2060, dummy software module 2710 is indistinguishable to the rest of simulation program 2086 from a fully functional software module simulating a component of target system 2010. That is, dummy software module 2710 forms states that are

accessible to the rest of simulation program 2086 as if dummy software module 2710 was actually performing the logical functions of the components being verified the verification engine 2060. Also in like relationship, the simulation program 2086 is inactive while the verification engine 2060 is active and the verification engine 2060 is inactive while the simulation program is active. When dummy hardware module 2072 notes a change in its state, it sends an interrupt to the system controller 2112, which stops the operation of reconfigurable interconnect assembly 2110 and subsequently sends the contents of dummy hardware module 2072 to workstation 2114, which houses the server process 2117. The server process places this received data into a table, which it then sends to simulation host computer 2118.

Dummy software module 2710 accepts the new state dictated by the new input, these new states result in activity in the rest of the program 2086, which is permitted to run until the state of dummy software module 2710 changes again, at which point dummy software module 2710 stops the execution of simulation program 2086, sends the code out from a socket of computer 2118. The code is placed into a table in the server 2117 that mirrors the storage buffer 2073 that is part of dummy module 2072. To accomplish this result, the server 2117, before the beginning of verification engine operation must construct a translation table translating each time position in the signal it receives from the simulation host computer 2118 into a table position. Likewise, when the server 2117 receives data from the verification engine 2060 it uses the translation table to translate from table position to time position on the data lines leading to the input/output socket of computer 2118. The translation table is computed from the configuration file that is produced by the GUI 2115 at the

same time the time position of the parameters sent to and from dummy software module 2710 is arbitrarily set.

As noted earlier, an embedded software development computer 2800 hosts an embedded design environment (EDE) 2810 that includes a debug tool 2812. The embedded design environment (EDE) may be from TASKING, which may be contacted over the internet at www.tasking.com. The EDE 2810 would typically be used with a ROM monitor, such as CrossView Pro, also available from TASKING. The ROM monitor resides in the 8051-memory area and permits a user to use the EDE 2810 to introduce breakpoints into the code that is executing on the 8051. When a breakpoint is encountered, user selected memory contents or CPU register contents are delivered to the user. The ROM monitor works according to the convention that the first line of code that it receives over the UART port indicates a memory location for the first line of code that follows. Therefore, this memory location would be set to the desired break point and would point to the break point handling routine in the ROM monitor. In this manner a user seated at the display screen of computer 2118 could set break points in the operation of the 8051-microcontroller program and periodically inspect the values of the various variables defined in this program.

A set of input/output pin toggle registers 2310 each contain a bit for every active input/output pin on a corresponding VC. Each toggle register bit changes from a ZERO to a ONE when the corresponding input/output pin switches state (is toggled). All of these bits are ANDED together by toggle ANDing device 2312, so that when the result of this AND operation equals a ONE, there is a positive indication that every input/output pin of the system being tested has been exercised. In addition, device 2312 has a reporting capability for forwarding to host

workstation 2114, the identities of all pins that have not been toggled, thereby aiding the verification engineer in completing his testing.

XI VERIFICATION ENGINE HAVING A BEHAVIORAL MODELLING CARD

FIG. 21 shows a block diagram of a fictitious target system in the form of an actuating assembly 3010 that includes a motor controller 3012 and a three phase brushless DC motor assembly 3014 Assembly 3010 is presented solely for purposes of clearly explaining the invention, which has to do with a method for verification testing of an assembly, such as assembly 3010.

The motor assembly 3014 includes a motor 3016 and a variable load 3018. In practice, the motor assembly 3014 could be positioning the arm of a robot. The load would vary according to the object, if any, that the robot was lifting. The rotor position of motor 3016 is detected by a rotor position encoder 3020, which is electrically connected to motor controller 3012. The motor 3016 is driven by the output of a three phase switching regulator 3022.

The motor controller 3012 accepts input from a position command input device 3030. In the robot example, this device would be an actuator permitting a person to command the robot arm position. The position commanded by way of device 3030 is fed into a microcontroller 3032, which compares the commanded position with the present position and computes a velocity and acceleration schedule to move the rotor from the present position to the desired position. The microcontroller 3032 includes a microprocessor 3034 and a memory 3036 that hosts a program for performing the motor control tasks. A switching regulator input conversion box 3040 accepts commutation and velocity correction commands and determines and sends a series of pulses that actuate the

transistors of switching regulator 3022, to achieve the velocity correction and commutation functions. A rotor position code formatting circuit 3042 accepts the rotor position code from encoder 3020 and places them into a form that is acceptable to microprocessor 3034.

Referring to FIG. 22, a verification system 3110 adapted to verify the operation of actuating assembly 3010 includes an interconnect assembly 3112 that may have a structure similar to any one that is shown in preceding figures of this application. Interconnect assembly 3112 connects a microprocessor card 3114, a central memory card 3116 that bears the motor control program that memory 3036 bears in the target system, a rotor position code formatting circuit hardware model 3118 and a rotor position code formatting circuit hardware model 3120. The hardware models 3118 and 3120 would likely each be implemented on a field programmable gate array (FPGA) as described in the previous applications. Also connected to interconnect assembly 3112 is a behavioral model card 3130 that includes a picoprocessor 3132, Java Picoprocessor available from both Sun Microsystems and Hewlett-Packard, both of Palo Alto, California, and a memory unit 3134, bearing a program that enables picoprocessor 3132 to mimic on an input/output basis the behavior of electric motor 3014.

There are a number of advantages of modeling an electric motor or other electro-mechanical device rather than using an actual electro-mechanical device. First, although there is not much of a safety issue in the example of an electric motor, there may be many instances in which it would be quite dangerous to verify software on the actual electro-mechanical device that it is designed to control. For example, software that is to be used in the control of a nuclear power plant is best verified in a test environment

to avoid a nuclear accident. Second, verification system 3110 may operate at a speed that is considerably slower than motor controller 3012. As such, it may not be able to send out control parameters at the time instances needed to accurately mimic in real time the operation of motor controller 3012. The rotor of motor 3016, however, accelerates and decelerates in response to a given set of inputs according to immutable physical laws. In contrast, time may be slowed down for a behavioral model without any loss in accuracy. Another advantage to the use of a behavioral model for an electro-mechanical device is that a single behavioral model card can mimic the behavior of any one of many different electro-mechanical devices, according to the program stored in memory unit 3134. A library of behavioral models is available from Synopsys, Inc. of Mountain View, California.

Similar to hardware model cards 3118 and 3120, behavioral model card 3130 includes an input/output pin for every input/output pin of the target system motor assembly 3114, which it is emulating. Because of this, there is no good to pack and unpack input/output signals onto a narrow bus.

Unlike hardware models 3118 and 3120, which respond to a set of inputs in a period of time determined by the number and durations of a set of gate delays, behavioral model card 3130 typically executes a significant number of instructions before its output changes state. In the case in which interconnect assembly 3112 is a time multiplexed bus as disclosed in incorporated application Serial No. 09/336,485, card 3130 has a period of time equal to the periodicity of the time multiplexed bus system clock to compute its output for a new set of inputs. As this type of system clock would typically be run at a frequency in the

MHz range and the picoprocessor 3132 would typically be run at a frequency in 100 MHZ range this generally would provide adequate time for a simple behavioral model to produce an output. Behavioral model card is equipped with an output not ready pin, however, which is routed to an interconnect assembly pause pin in the interconnect assembly controller 3113. In the case in which the interconnect assembly 3112 is a time multiplexed bus, this would cause the bus controller to pause the system clock until the output of the behavioral model card 3112 was ready.

Additionally a simulation host computer 3140 running a program simulating other parts of the electromechanical system provides load inputs for electrical motor behavioral model card 3130 and a storage device provides a user input set of vectors 3142.

FIG. 23 shows an example of a purely communicative and generally electrical (portions could be optical) device that may require verification. This is a router 3210 of the type used to route messages over an Internet or the Internet. To be properly routed by router 3210 any message from an attached network 3212 must be formatted to a particular standard. Within the scope of this standard, however, there are nine permitted formatting standards, each of which must be handled somewhat differently. For this reason there are nine different format routing blocks 3214-3230 in router 3210. An initial routing 3234 block routes each message to the correct format routing block and a packing and final routing block 3236 packs together messages going to the same location.

FIG. 24 shows a verification engine configured to verify the operation of router 3210. The reference numbers of FIG. 24 correspond to the target system reference numbers of FIG. 23, with each hardware model bearing a

reference number of the portion of router 3210 being modeled plus one hundred. If one were to design a router 3210 in the form of a system-on-a-chip, it would be important to carefully verify the circuit design prior to etching it into silicon. One way of doing this is to construct a hardware model 3312-3326 and 3334 for each routing block 3212-3226 and 3234 and connect all of the hardware models together, in the manner taught in the incorporated applications, by an interconnect assembly 3340. In the example presented in FIG. 24, format 9 routing block 3240 is still in the process of being designed, and accordingly is simulated in a simulation host computer 3330. Alternatively, as shown for the format 8 routing block 3228, the routing block can be modeled by a behavioral model card 3328, having an input/output pin for each input/output pin of the target system format 9 (nine) routing block 3230. One difficulty that arises in the verification process for circuitry of this kind is that of accurately modeling the environment to which router 3210 is to be connected. Of course, one cannot connect the circuitry undergoing verification to the Internet. Therefore, the verification engine on FIG. 24 includes an Internet behavioral model 3342 providing the complete range of Internet traffic in order to thoroughly test out the target system being verified. In a preferred embodiment, behavioral model 3342 includes means for receiving user input, for example, a jack connected to a keyboard, thereby permitting a user to exercise a measure of control over the data packets being sent to interconnect assembly 3340.

In addition, the Internet or network model 3342 must respond to messages routed through the router model 3310 in the same manner in which an actual network would in order to accurately test router model 3310. If a simulation

hosting computer hosts a behavioral model of a network this will greatly slow the verification process in the manner described in the background of the invention section. Accordingly, it is highly desirable to host a behavioral model on a card that is directly attached to the interconnect assembly and that does not primarily host a simulator, which would typically necessitate that all inputs be packed into and unpacked from a bus. The behavioral model program in memory 3346 causes microprocessor or picoprocessor 3348 to imitate the operation of the Internet. The connection between the behavioral model card 3342 and the interconnect assembly, unhampered as it is by the need to pack and unpack data from a bus, is termed a dedicated port.

As used in this application, the term "sub-portion" may include all of a "portion."

As used in this application, the term "pin" means any sort of electrical lead.

The term "set" is used in the mathematical sense in this application and includes a set that includes only one element.

The terms and expressions which have been employed in the foregoing specification are used therein as terms of description and not of limitation, and there is no intention, in the use of such terms and expressions, of excluding equivalents of the features shown and described or portions thereof, it being recognized that the scope of the invention is defined and limited only by the claims which follow.

CLAIMS

1. A verification engine for verifying the design of a target system having a plurality of components interconnected by a plurality of target system buses, said verification engine comprising:

(a) a first hardware model and a second hardware model, both configured as a said component and having a set of hardware model input/output pins;

(b) a first bus wrapper connected to said first hardware model and a second bus wrapper connected to said second hardware model;

(c) a set of bus lines, each said bus line being connected to said first bus wrapper and said second bus wrapper;

(d) wherein each said bus wrapper further has switchable communicative circuitry that switchably communicatively connects each said hardware model input/output pin to a bus line and has a control block controlling said switchable communicative circuitry;

(e) a system controller connected to at least some of said bus lines and adapted to transmit a sequence of time synchronization information to each said bus wrapper control block, said time synchronization information sufficient to permit said control blocks to uniformly determine a time slot number; and

(f) wherein said control blocks uniformly determine said time slot number in response to said time synchronization information and in response thereto each control block switches at least one said input/output pin

into communicative contact with a said bus line so that at least one said input/output pin from said first hardware model is connected to a said input/output pin of said second hardware model.

2. The verification engine of claim 1 including additional hardware models each connected to an additional bus wrapper which is connected to said set of bus lines.

3. The verification engine of claim 1 wherein said system controller transmits one out of a set of target system clock information during each time slot number and said first hardware model has a first target system clock input pin and said first bus wrapper is capable of generating a clock input and sending it to said first target system clock input pin in response to a first target system clock indicating signal.

4. The verification engine of claim 3 wherein said set of target system clock information includes a null command which indicates that no said target system clock signal should be sent by any of said bus wrappers.

5. The verification engine of claim 3 wherein said second hardware model includes a second target system clock input pin and said second bus wrapper further includes a second bus wrapper target system clock output pin connected to said second target system clock input pin and wherein said second bus wrapper determines, on each said time slot, based on said target system clock indicating signal whether to send a target system clock signal over said second bus wrapper target system clock output pin.

6. The verification engine of claim 5 wherein said set of target system clock signals includes a first clock indicating signal and a second clock indicating signal and wherein said first bus wrapper sends a target system clock over said first bus wrapper target system clock output pin in response to said first clock indicating signal and said second bus wrapper sends a target system clock over said second bus wrapper target system clock output pin in response to said second clock indicating signal.

7. The verification engine of claim 3 wherein said set of target system clock signals further includes a second clock indicating signal and said first hardware model further includes a second target system clock input pin and said first bus wrapper is further capable of generating a second clock input and sending it to said second target system clock input pin in response to a second target system clock indicating signal.

8. The verification engine of claim 1 in which said time synchronization information includes a time slot number that is transmitted in a plurality of time slots.

9. A verification engine for verifying the design of a target system having a plurality of components interconnected by a plurality of target system buses, said verification engine comprising:

(a) a plurality of reconfigurable bus wrappers, each having a bus wrapper/hardware model set of pins and a bus wrapper/bus line set of pins, switchable communicative circuitry that switchably communicatively connects each said

bus wrapper/hardware model pin to a bus wrapper/bus line pin and a control block controlling said switchable communicative circuitry;

(b) a set of bus lines, each said bus line having a bus line/bus wrapper pin for each bus wrapper, each said bus line/bus wrapper pin, being connected to a bus wrapper/bus line pin;

(c) a system controller connected to at least some of said bus lines and adapted to transmit time synchronization information sufficient for said control blocks to uniformly determine a time slot number; and

(d) wherein, responsive to said time synchronization information, each said control block determines said time slot number and responsive to a predetermined one of said time slot numbers, switches at least one bus wrapper/hardware model pin into communicative contact with a said bus wrapper/bus line pin.

10. The verification engine of claim 9 wherein said time synchronization information includes a time slot number.

11. A method of determining drive direction between a first circuit node that is communicatively connected to a first driver input pin of a first driver and a second circuit node that is communicatively connected to a second driver input pin of a second driver, said first driver and said second driver being connected by a

connecting line, and wherein said first and second circuit nodes are tentatively commanded to be connected by enabling either said first driver or said second driver with input from said first circuit node or said second circuit node respectively, said method comprising:

(a) automatically forming a first test result by determining if said first circuit node is being driven; and

(b) recording said first test result in a first format.

12. The method of determining drive direction of claim 11, further including the steps of:

(a) automatically forming a second test result by determining if said second circuit node is being driven;

(b) recording said second test result in said first format;

(c) comparing said first test result to said second test result;

(d) if said first test result is different from said second test result, using said first driver or said second driver to drive said interconnecting line from whichever of said circuit nodes is being driven.

13. The method of claim 12 wherein if both said first circuit node and said second circuit node are being driven, this is recorded as a fight condition and if neither one of said first circuit node and said second circuit node is being driven this is recorded as a float condition.

14. The method of claim 12 wherein said fight condition causes a cancellation of said tentative command for connection of said first and second nodes.

15. The method of claim 12 wherein said float condition causes an interrupt to system operation.

16. The method of claim 12 wherein said first node encompasses a first conductive link between a first hardware model and a first bus wrapper and said second node encompasses a second conductive link between a second hardware model and a second bus wrapper and said interconnecting line is a bus line connecting said first bus wrapper to said second bus wrapper.

17. The method of claim 12 wherein said first node is one out of a multiplicity of first nodes, each encompassing a conductive link between said first hardware model and a first bus wrapper that is part of a verification engine and wherein said second node is one of a multiplicity of second nodes, each encompassing a conductive link between said second hardware model and said second bus wrapper in said verification engine, and said bus line is one out of multiplicity of bus lines connecting said first bus wrapper to said second bus wrapper.

18. The method of claim 13 wherein said verification engine further includes a bus fight reporting table that specifies for each bus line at least one binary value indicating whether a prospective bus fight on said bus line is to result in an interruption of verification engine operation.

19. The method of claim 13 wherein said verification engine further includes a bus float reporting table that specifies for each bus line at least one binary value indicating whether a prospective bus float on said bus line is to result in an interruption of verification engine operation.

20. The method of claim 18 wherein said bus lines are time multiplexed into time slots and in which said bus fight reporting table specifies for each bus line a binary value for each time slot indicating whether a prospective bus fight on said bus line in said time slot is to result in an interruption of verification engine operation.

21. The method of claim 19 wherein operation of said bus lines is time multiplexed into time slots and wherein said bus float reporting table specifies for each bus line, for each time slot, a binary value indicating whether a prospective bus float on said bus line in said time slot will result in an interruption of verification engine operation.

22. A bus adapted to selectively couple a first electrical component having a set of first component input/output pins to a second electrical component having a set of second component input/output pins, comprising:

(a) a first bus wrapper having:

(i) a set of first bus wrapper-to-component pins adapted to be connected to said first component input/output pins;

(ii) a set of first bus wrapper bus line pins;

(iii) a set of latches, each said latch being switchably communicatively connected to a said first bus wrapper-to-component pin and to a said first bus wrapper bus line pin; and

(iv) a local condition sensor;

(b) a second bus wrapper having:

(i) a set of second bus wrapper-to-component pins adapted to be connected to said second component input/output pins;

(ii) a set of second bus wrapper bus line pins; and

(iii) a set of latches, each said latch being switchably communicatively connected to a said second bus wrapper-to-component pin and to a said second bus wrapper bus line pin;

(c) a set of bus line conductors connecting each said first bus wrapper bus line pin to a said second bus wrapper bus line pin; and

(d) a system controller having a set of system controller bus line pins connected to said set of bus line conductors and adapted to transmit to said bus wrappers time synchronization information sufficient for said bus wrappers to uniformly determine a time slot number; and

(e) wherein each bus wrapper is adapted to determine, for each said time slot number which, if any, of its said latches is active during said time slot and in which said first bus wrapper is further adapted to determine,

at least in part by said local condition sensor and said time slot number, which said active latches will be switched into a transmitting communicative coupling with a said first bus wrapper-to-component pin and which will be switched into a transmitting communicative coupling with a said first bus wrapper bus line pin.

23. The bus of claim 22 wherein said local condition sensor senses whether a first input/output pin of said first electrical components is driving.

24. A method for verification testing of a target system design made up of a plurality of components connected by a plurality of buses that permit a defined flow of data between said components, said method comprising:

(a) connecting hardware models of at least some of said components with a time multiplexed bus; and

(b) controlling said time multiplexed bus to permit said defined flow of data between said hardware models.

25. An apparatus for interfacing to an integrated circuit that contains bi-directional pins, said apparatus comprising:

(a) a soft drive adapted to soft drive a said bi-directional pin high during a high drive time and to soft drive said bi-directional pin low during a low drive time;

(b) a first flip-flop configured to save the output of said bi-directional pin during its high drive time;

(c) a second flip-flop configured to save the output of said bi-directional pin during its low drive time; and

(d) an Exclusive OR gate having a first input that is connected to said output of said first flip-flop and a second input that is connected to said output of said second flip-flop and having an output which, after said high and low drive times, is thereby indicative of pin drive direction.

26. The apparatus of claim 25, further including a bi-directional buffer for said bi-directional pin and wherein said bi-directional buffer changes drive direction in response to said output of said Exclusive OR gate.

27. A verification engine adapted to model a target system having a set of components, said verification engine comprising:

(a) a multiplicity of hardware models, each hardware model representing a said component;

(b) a bus linking together said hardware models and including a system controller;

(c) a computer running a simulation program of a said component that is not represented by a said hardware model and is thereby a simulated component, said computer further being communicatively connected to said system controller.

28. A device for facilitating the prototyping of a target system electronic design by a developer, said target system electronic design including a first component, having an internal construction which may be unknown to said developer and having a first pin, a second component and a third component that transmits a first signal to said first pin of said first component, said device comprising:

(a) a computer including a socket and a memory assembly bearing a program written in a simulation language and adapted to simulate said third component and to form a first model state that models said first signal;

(b) a computer software object, stored in said memory of said computer and being further adapted to translate said first model state into a first socket command thereby causing a first socket output signal, which models said first signal, to be transmitted from said

socket;

(c) a communicative assembly being operatively coupled to said socket;

(d) a first hardware model adapted to model said first component and including a first model pin that models said first pin, said first model pin being operatively connected to said communicative assembly; and

(e) a second hardware model adapted to model said second component and being operatively connected to said communicative assembly.

29. The device of claim 28 wherein said first hardware model is physically identical to said first component.

30. The device of claim 28 wherein said first hardware model comprises an FPGA configured to implement the circuitry of the first component.

31. A device for facilitating the prototyping of a target system electronic design by a developer, said target system electronic design including a first component, having an internal construction which may be unknown to said developer and having a first pin, a second component and a third component that receives a first signal from said first pin of said first component and forms a first state in response to said first signal, said device comprising:

(a) a communicative assembly;

(b) a first hardware model adapted to model said first component and including a first model pin that models said first pin, said first model pin being operatively connected to said

communicative assembly, said first hardware model being further adapted to transmit said first signal from said first pin;

(c) a second hardware model adapted to model said second component and being operatively connected to said communicative assembly; and

(d) a computer including a socket and a memory assembly bearing a program written in a simulation language and adapted to simulate said third component and to receive a first socket signal over said socket, said socket being operatively connected to said communicative assembly;

(e) a computer software object, stored in said memory of said computer and being adapted to model said third component and to receive said first socket signal and to form a first model state in response to said first socket signal said first model state modeling said first state.

32. A device for facilitating the prototyping of a target system electronic design by a developer, said electronic design including a microprocessor, a first memory assembly bearing a program adapted to run on said microprocessor, a component, having an internal construction which may be unknown to said developer and a first communicative linking assembly connecting said microprocessor to said computer memory and said component, said device comprising:

(a) a hardware implemented microprocessor model adapted to model said microprocessor at least in part;

(b) a hardware component model adapted to model said component;

(c) a second memory assembly bearing said program and memory contents;

(d) a model communicative assembly connecting said model memory assembly and said component model to said microprocessor model;

(e) a computer;

(f) a linking and reporting assembly, linking said computer to said model communicative assembly and adapted to deliver selected ones of said memory contents to said computer at developer specified points in said program.

33. The device of claim 32 wherein said microprocessor model physically comprises said microprocessor.

34. A verification engine for testing a set of components, each component having a set of input/output pins, said verification engine having the ability to measure toggle coverage for a set of component pins and comprising:

(a) an interconnective assembly having a toggle detector for each said input/output pin;

(b) memory space for recording the output of each toggle detector;

(c) a detecting and reporting assembly for detecting and reporting to a user which, if any, of said input/output pins has not been toggled.

35. A method of mapping a system level hardware description, containing a set of target system components,

each said target system component designated by a target system component name in said system level hardware description, into a set of virtual components, each present in a library of hardware models and being designated by a virtual component name, comprising the steps of:

(a) parsing said system level hardware description into said set of target system components and displaying a said target system component name representative of a said target system component;

(b) displaying a said virtual component name;

(c) permitting a user to match said target system component name to said virtual component name thereby designating said hardware model associated with said virtual component name for said target system component.

36. The method of claim 35 wherein a plurality of said virtual component names are displayed simultaneously.

37. The method of claim 35 wherein a plurality of said target system component names are displayed simultaneously.

38. A method of partitioning a system level hardware description, containing a set of target system components, each said target system component designated by a target system component name in said system level hardware description, into a first subset of components that are to be modeled in a verification engine and a second subset of components that are to be modeled in a simulator, comprising the steps of:

(a) parsing said system level hardware

description into said set of target system components and displaying a said target system component name;

(b) permitting a user to designate said target system component corresponding to said target system component name as belonging to said first subset of components or said second set of components; and

(c) creating an interconnect net list between said first set of components and said second set of components.

39. The method of claim 38 wherein a plurality of target system component names are displayed simultaneously.

40. The method of claim 38 wherein said target system component corresponding to said target system component name is automatically designated as belonging to said first subset of components unless said user specifically indicates that said target system component should belong to said second subset of components.

41. A verification engine for verifying the design of a target system having a first component a second component and a further portion, including a sub-portion having a set of sub-portion input/output pins, said verification engine comprising:

(a) a first hardware model and a second hardware model, configured as said first component and said second component respectively;

(b) an interconnect assembly communicatively linking said first hardware model and said second hardware model together; and

(c) a processor assembly including:

(i) a set of processor assembly input/output pins connected to said interconnect assembly; and

(ii) a memory assembly bearing a program that is adapted to create a behavioral model of said sub-portion when executed on said processor assembly and further adapted to control said set of processor assembly input/output pins to respond to stimulus as said sub-portion input/output pins respond during target system operation.

42. The verification engine of claim 41 in which said reconfigurable interconnect assembly comprises a time multiplexed bus.

43.The verification engine of claim 1 further including a computer hosting an executing simulation program, which is connected to said reconfigurable interconnect assembly.

44. A method for verifying the design of a target system having a first component a second component and a further portion, including a sub-portion, said method comprising the steps of :

(a) providing a first hardware model and a second hardware model, configured as said first component and said second component respectively;

(b) providing an interconnect assembly communicatively linking said first hardware

model and said second hardware model together;

(c) providing a processor assembly connected to said interconnect assembly;

(d) operating said first hardware model and said second hardware model, thereby creating interconnect assembly signals and making said interconnect assembly signals available to said processor assembly;

(e) executing a program on said processor, said program performing the steps of:

(i) establishing a cycle based behavioral model of said sub-portion; and

(ii) directly porting said interconnect assembly output signals to said behavioral model and directly porting said behavioral model output signals to said interconnect assembly.

45. The method of claim 44 in which said reconfigurable interconnect assembly comprises a time multiplexed bus.

46. The method of claim 44 further including providing a computer hosting an executing simulation program, which is connected to said reconfigurable interconnect assembly.

47. A verification engine for verifying the design of a target system having a first electrical component and a second electrical component and that is adapted to control an electro-mechanical device having a set of first

input/output pins and being operatively connected to said first and second electrical components:

(a) a first hardware model and a second hardware model, configured as said first component and said second component respectively;

(b) an interconnect assembly communicatively linking said first hardware model and said second hardware model together; and

(c) a processor assembly including:

(i) a set of second input/output pins connected to said interconnect assembly; and

(ii) a memory assembly bearing a program that is adapted to create a behavioral model of said electro-mechanical device when executed on said processor assembly and to cause said set of second input/output pins to respond to stimulus as do said set of first input/output pins during target system operation.

48. A verification engine for verifying the design of a target system having a first electrical component, a second electrical component and a further portion, including a sub-portion, said verification engine comprising:

(a) a first hardware model and a second hardware model, configured as said first component and said second component respectively;

(b) an interconnect assembly communicatively linking said first hardware model and said second hardware model together;

(c) a system controller operatively connected to said interconnect assembly and transmitting a system clock signal to said first hardware model and said second hardware model by way of said interconnect assembly

(d) a processor assembly including:

(i) a set of input/output pins connected to said interconnect assembly;

(ii) a memory assembly bearing a program that is adapted to create a behavioral model when executed on said processor assembly;

(iii) an internal clock, electrically connected to a clock input of said microprocessor and that is faster than said system clock; and

(iv) wherein said processor assembly is adapted to respond to a set of inputs received from said interconnect assembly within a single system clock period.

49. The verification engine of claim 48 in which said reconfigurable interconnect assembly comprises a time multiplexed bus.

50. The verification engine of claim 48 further including a computer hosting an executing simulation program, which is connected to said reconfigurable interconnect assembly.

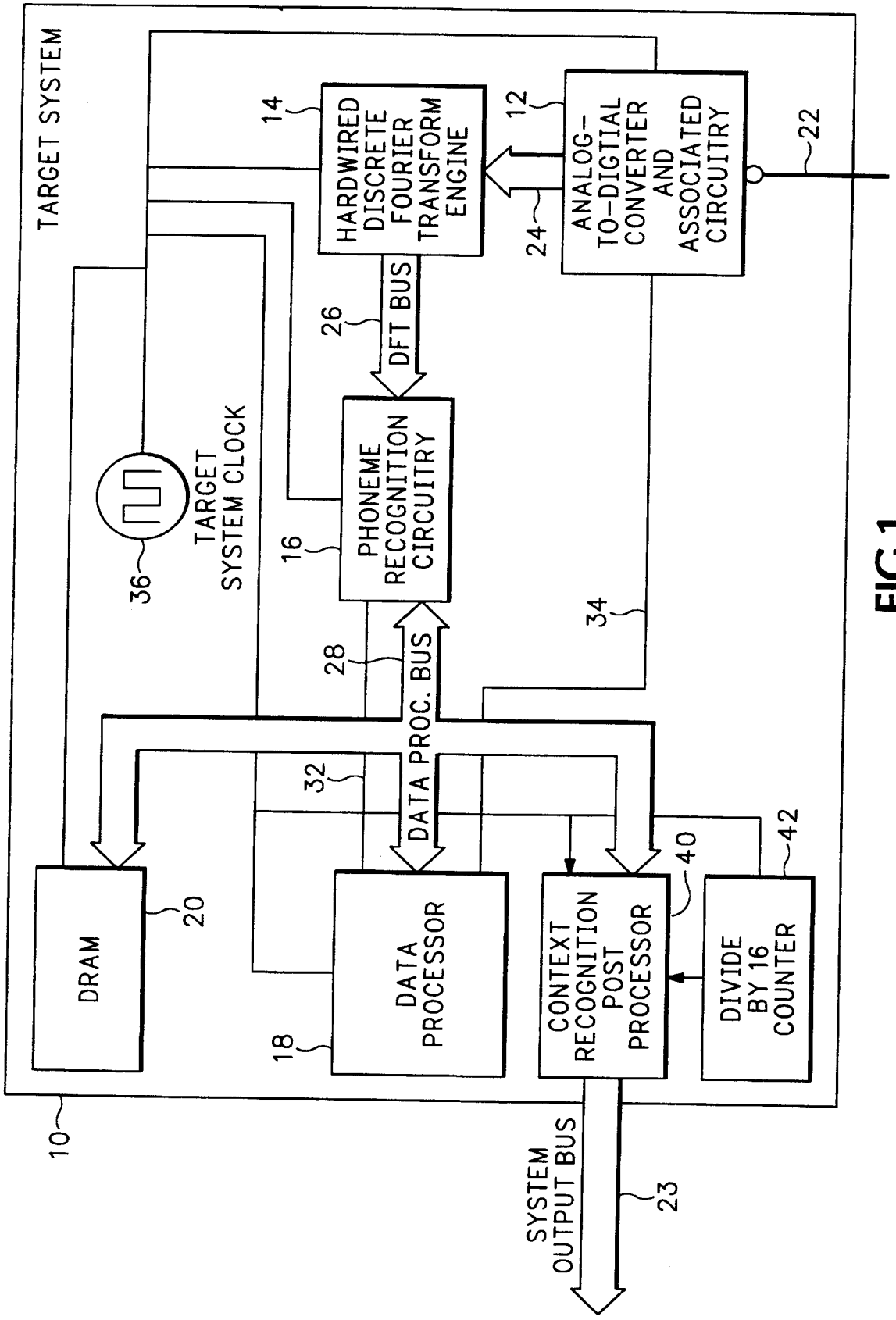

TARGET SYSTEM
HARDWIRED DISCRETE FOURIER TRANSFORM ENGINE
14
ANALOG-TO-DIGTIAL CONVERTER AND ASSOCIATED CIRCUITRY
12
22
24
DFT BUS
26
PHONEME RECOGNITION CIRCUITRY
16
TARGET SYSTEM CLOCK
36
34
28
DATA PROC. BUS
32
DRAM
20
DATA PROCESSOR
18
CONTEXT RECOGNITION POST PROCESSOR
40
DIVIDE BY 16 COUNTER
42
10
SYSTEM OUTPUT BUS
23

FIG.1

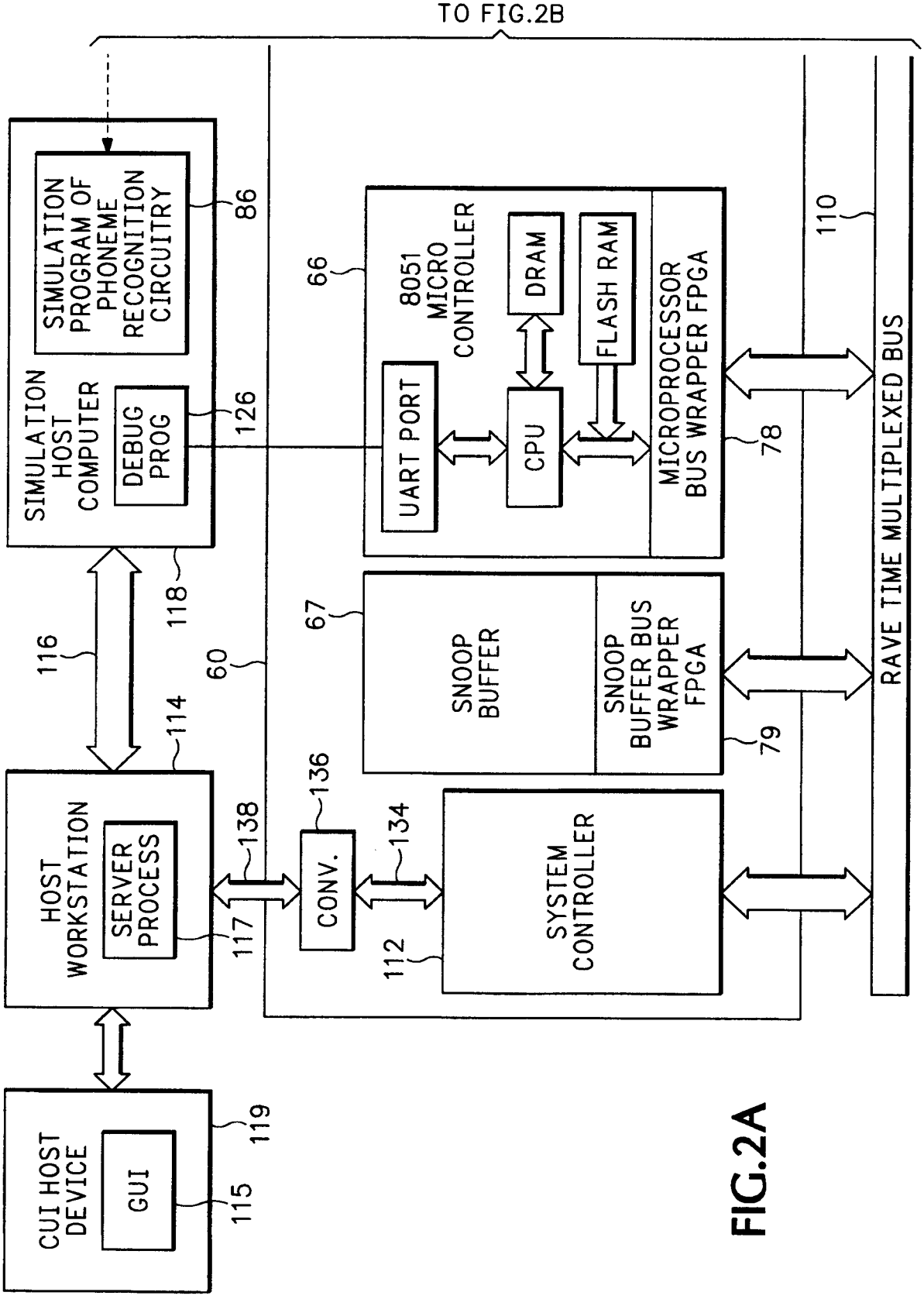

TO FIG.2B
SIMULATION PROGRAM OF PHONEME RECOGNITION CIRCUITRY
86
SIMULATION HOST COMPUTER
DEBUG PROG
126
66
8051 MICRO CONTROLLER
DRAM
FLASH RAM
UART PORT
CPU
MICROPROCESSOR BUS WRAPPER FPGA
78
110
RAVE TIME MULTIPLEXED BUS
116
118
60
67
SNOOP BUFFER
SNOOP BUFFER BUS WRAPPER FPGA
79
114
136
138
134
HOST WORKSTATION
SERVER PROCESS
117
CONV.
SYSTEM CONTROLLER
112
CUI HOST DEVICE
GUI
115
119

FIG.2A

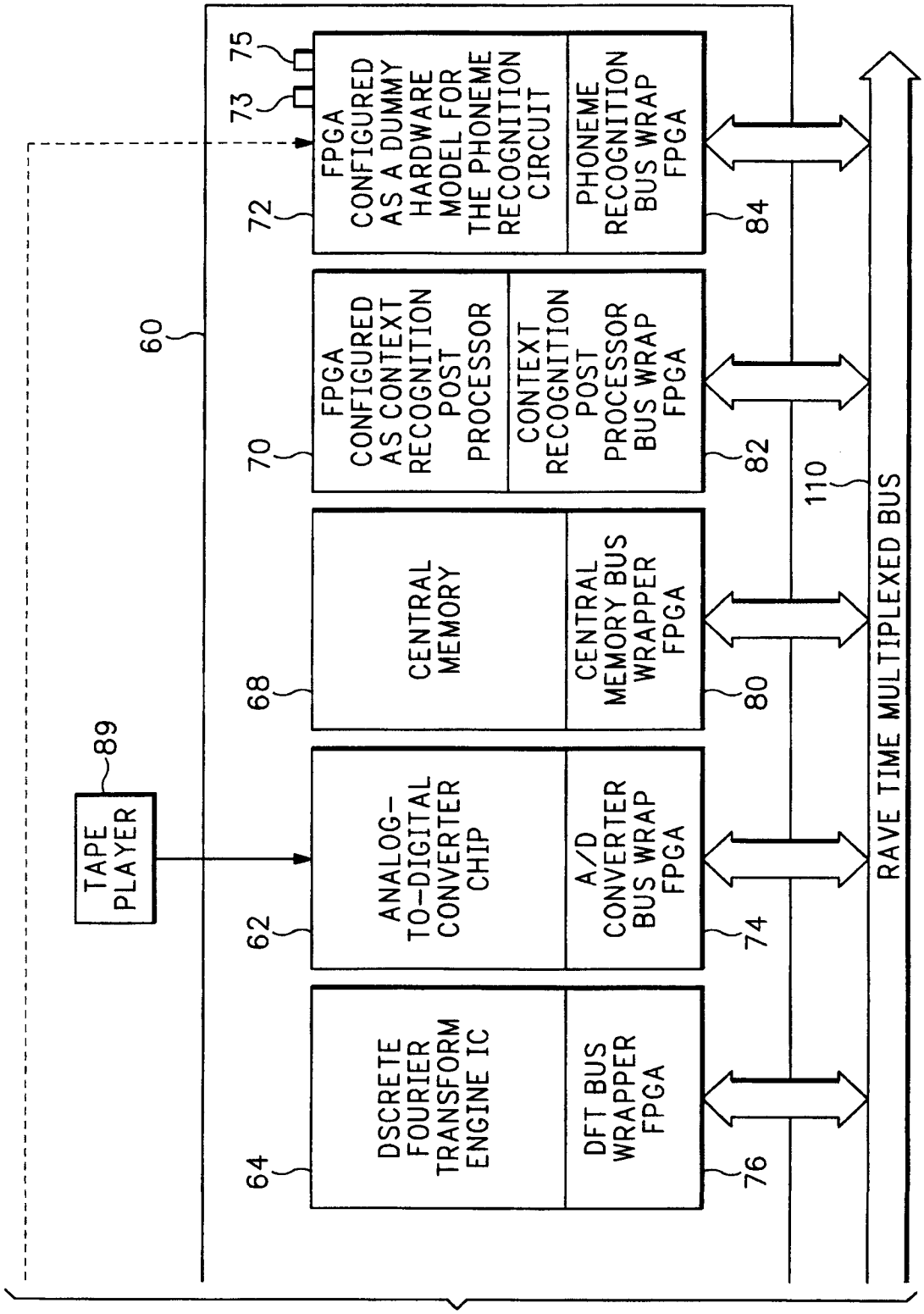

FPGA CONFIGURED AS A DUMMY HARDWARE MODEL FOR THE PHONEME RECOGNITION CIRCUIT
PHONEME RECOGNITION BUS WRAP FPGA
FPGA CONFIGURED AS CONTEXT RECOGNITION POST PROCESSOR
CONTEXT RECOGNITION POST PROCESSOR BUS WRAP FPGA
CENTRAL MEMORY
CENTRAL MEMORY BUS WRAPPER FPGA
ANALOG-TO-DIGITAL CONVERTER CHIP
A/D CONVERTER BUS WRAP FPGA
DSCRETE FOURIER TRANSFORM ENGINE IC
DFT BUS WRAPPER FPGA
TAPE PLAYER
RAVE TIME MULTIPLEXED BUS
60
62
64
68
70
72
73
74
75
76
80
82
84
89
110
FROM FIG.2A

FIG.2B

TIME SLOT #	BUS	BETWEEN	TARG. SYS. CLOCK DOMAIN	TAR. SYS. CLK ISS?
TIME SLOT 1	A/D OUTPUT BUS DFT BUS	ANALOG-TO-DIGITAL CONVERTER AND DISCRETE FOURIER TRANSFORM ENGINE; DISCRETE FOURIER TRANSFORM ENGINE AND PHONEME RECOGNITION CIRCUIT DUMMY CARD;	A	NO
TIME SLOT 2	DATA PROC. BUS	PHONEME RECOGNITION CIRCUIT DUMMY CARD; MICROPROCESSOR; CENTRAL MEMORY; CONTEXT RECOGNITION POST PROCESSOR;	A	NO
TIME SLOT 3	DISCRETE SIGNALS	MICROPROCESSOR; A/D CONVERTER SYSTEM CONTROLLER/WORKSTATION	A	1/1
TIME SLOT 4	OUTPUT BUS	CONTEXT RECOGNITION POST PROCESSOR; CENTRAL MEMORY	B	1/16

FIG.3

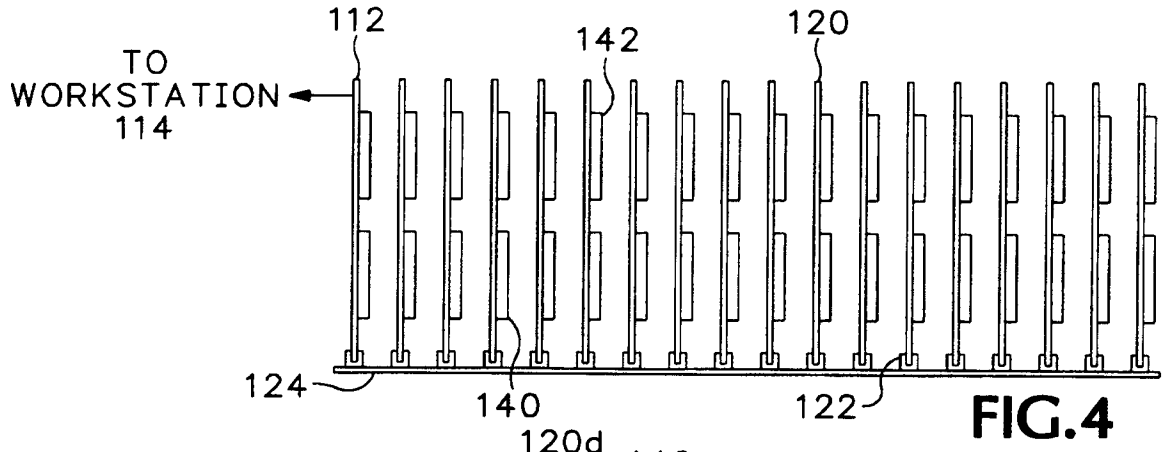


FIG.4

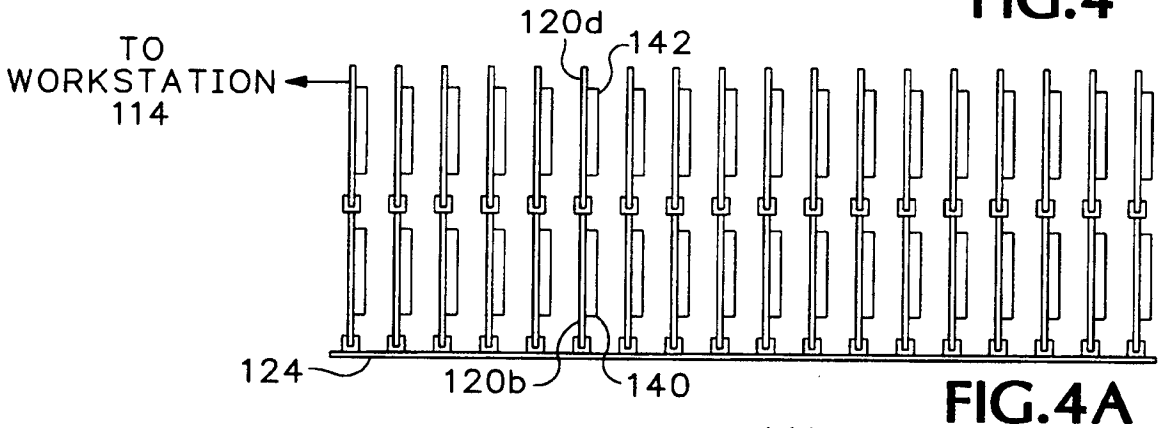


FIG.4A

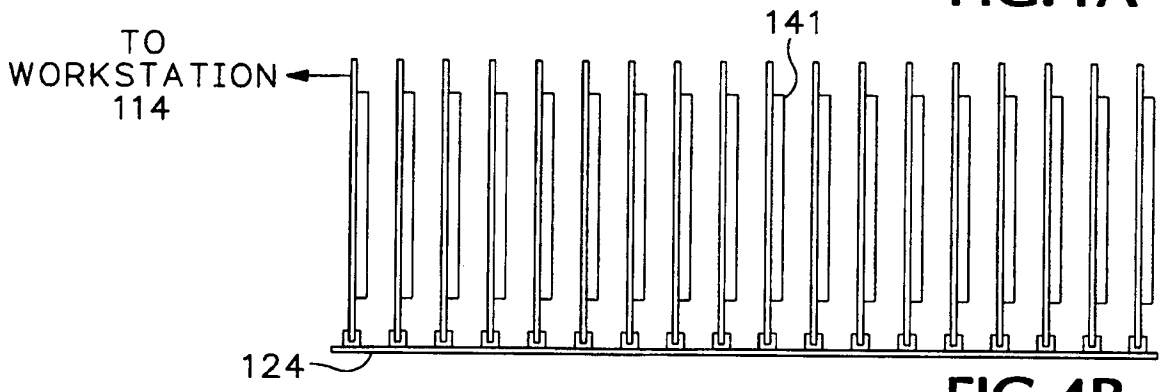


FIG.4B

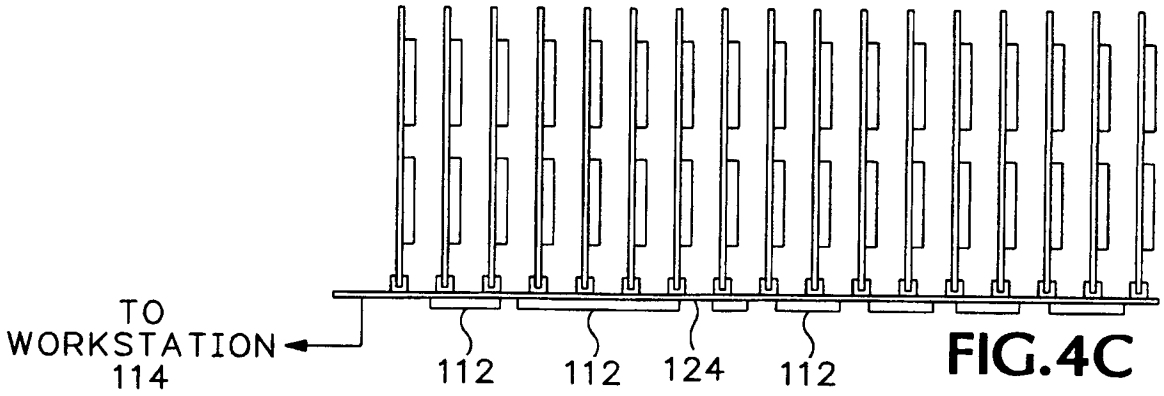


FIG.4C

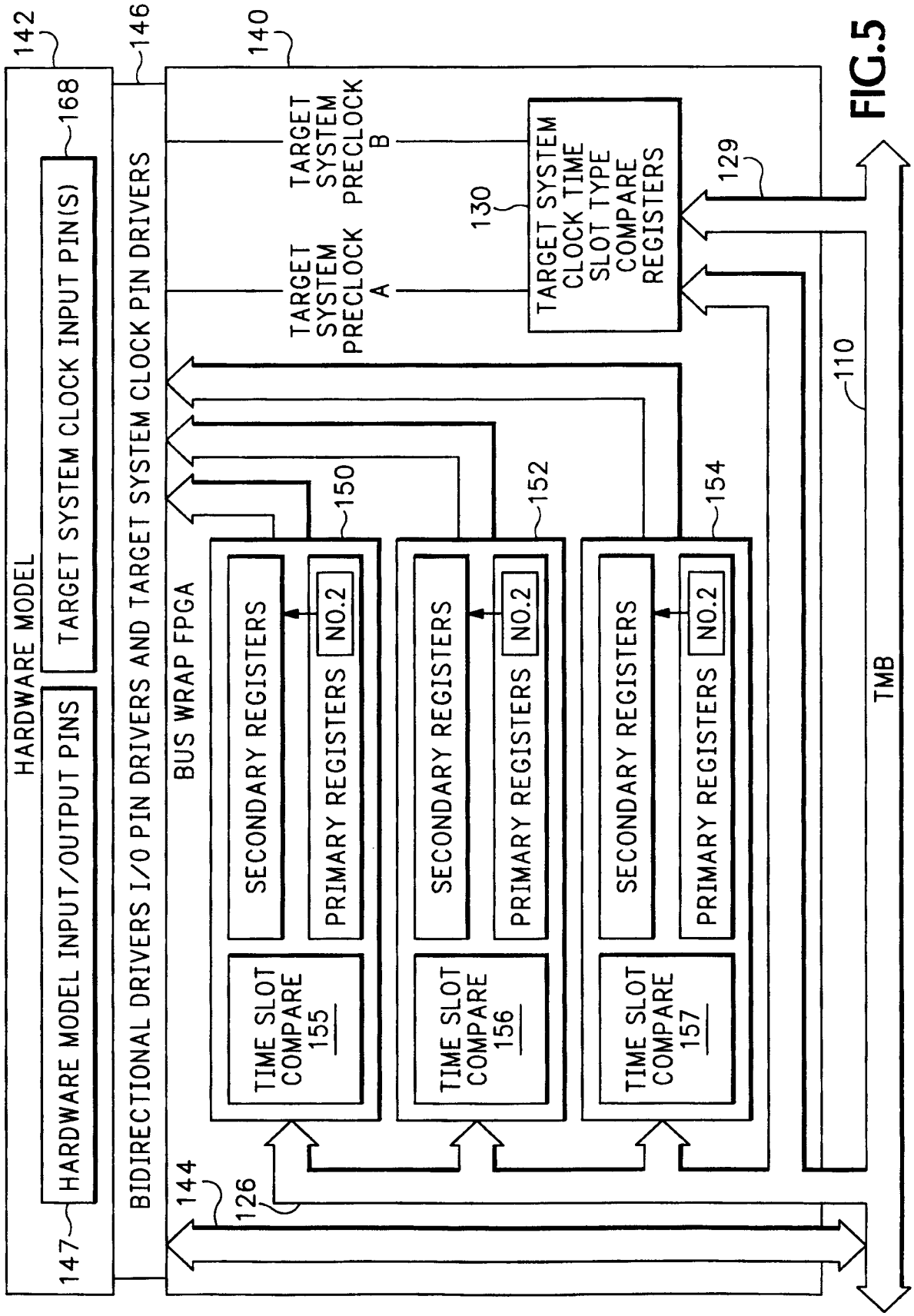

142
146
140
168
HARDWARE MODEL
TARGET SYSTEM CLOCK INPUT PIN(S)
HARDWARE MODEL INPUT/OUTPUT PINS
147
BIDIRECTIONAL DRIVERS I/O PIN DRIVERS AND TARGET SYSTEM CLOCK PIN DRIVERS
BUS WRAP FPGA
TARGET SYSTEM PRECLOCK A
TARGET SYSTEM PRECLOCK B
130
TARGET SYSTEM CLOCK TIME SLOT TYPE COMPARE REGISTERS
129
110
TMB
150
152
154
SECONDARY REGISTERS
PRIMARY REGISTERS
NO.2
TIME SLOT COMPARE 155
TIME SLOT COMPARE 156
TIME SLOT COMPARE 157
144
126

FIG.5

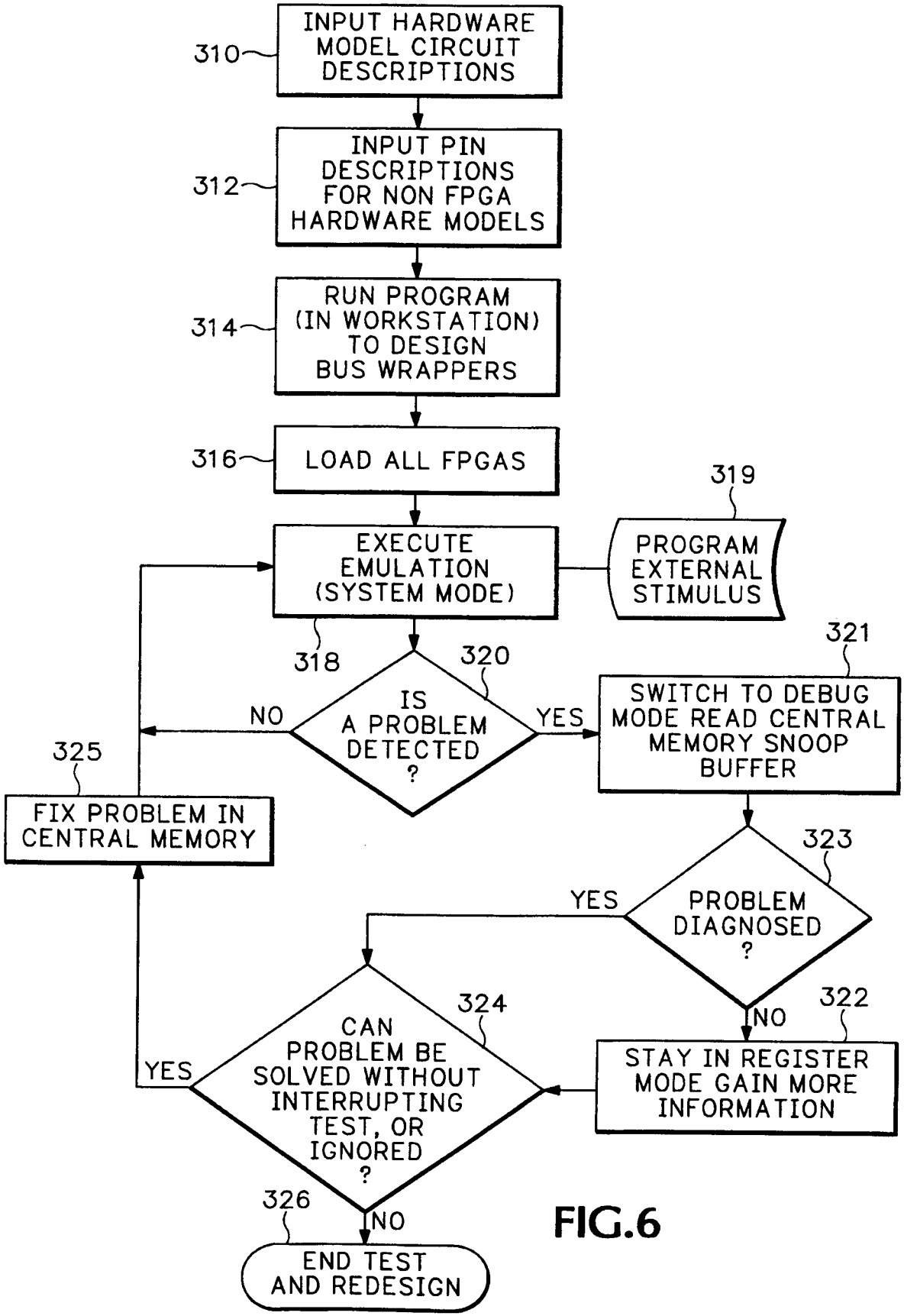

310
INPUT HARDWARE MODEL CIRCUIT DESCRIPTIONS
312
INPUT PIN DESCRIPTIONS FOR NON FPGA HARDWARE MODELS
314
RUN PROGRAM (IN WORKSTATION) TO DESIGN BUS WRAPPERS
316
LOAD ALL FPGAS
319
PROGRAM EXTERNAL STIMULUS
EXECUTE EMULATION (SYSTEM MODE)
318
320
IS A PROBLEM DETECTED ?
NO
YES
321
SWITCH TO DEBUG MODE READ CENTRAL MEMORY SNOOP BUFFER
325
FIX PROBLEM IN CENTRAL MEMORY
323
PROBLEM DIAGNOSED ?
YES
NO
322
STAY IN REGISTER MODE GAIN MORE INFORMATION
324
CAN PROBLEM BE SOLVED WITHOUT INTERRUPTING TEST, OR IGNORED ?
YES
326
NO
END TEST AND REDESIGN

FIG.6

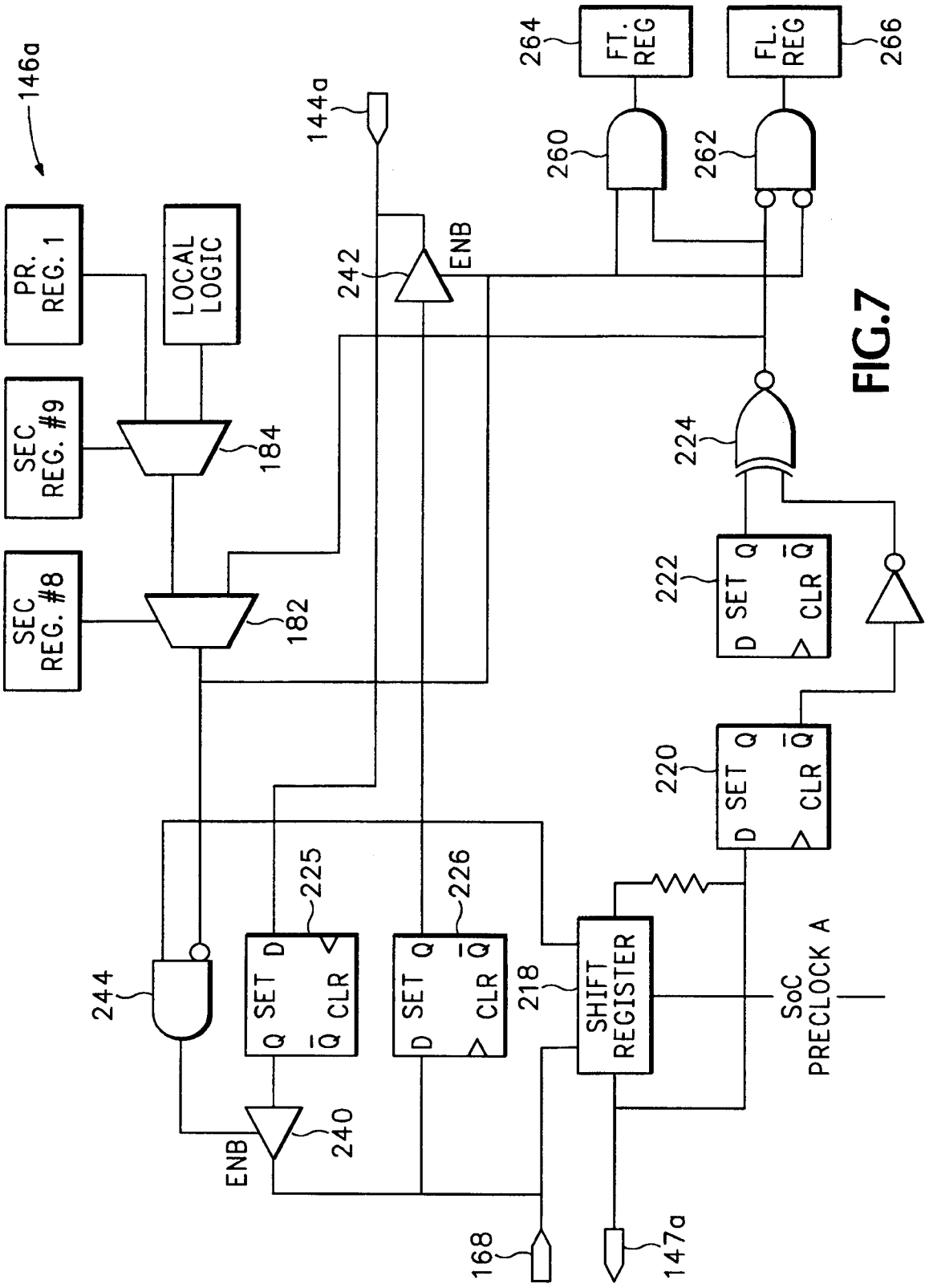

146a
PR. REG. 1
LOCAL LOGIC
SEC REG. #9
SEC REG. #8
184
182
144a
242
ENB
264
FT. REG
266
FL. REG
260
262
224
222
220
D SET Q
CLR Q
225
226
244
240
218
SHIFT REGISTER
SoC PRECLOCK A
168
147a

FIG.7

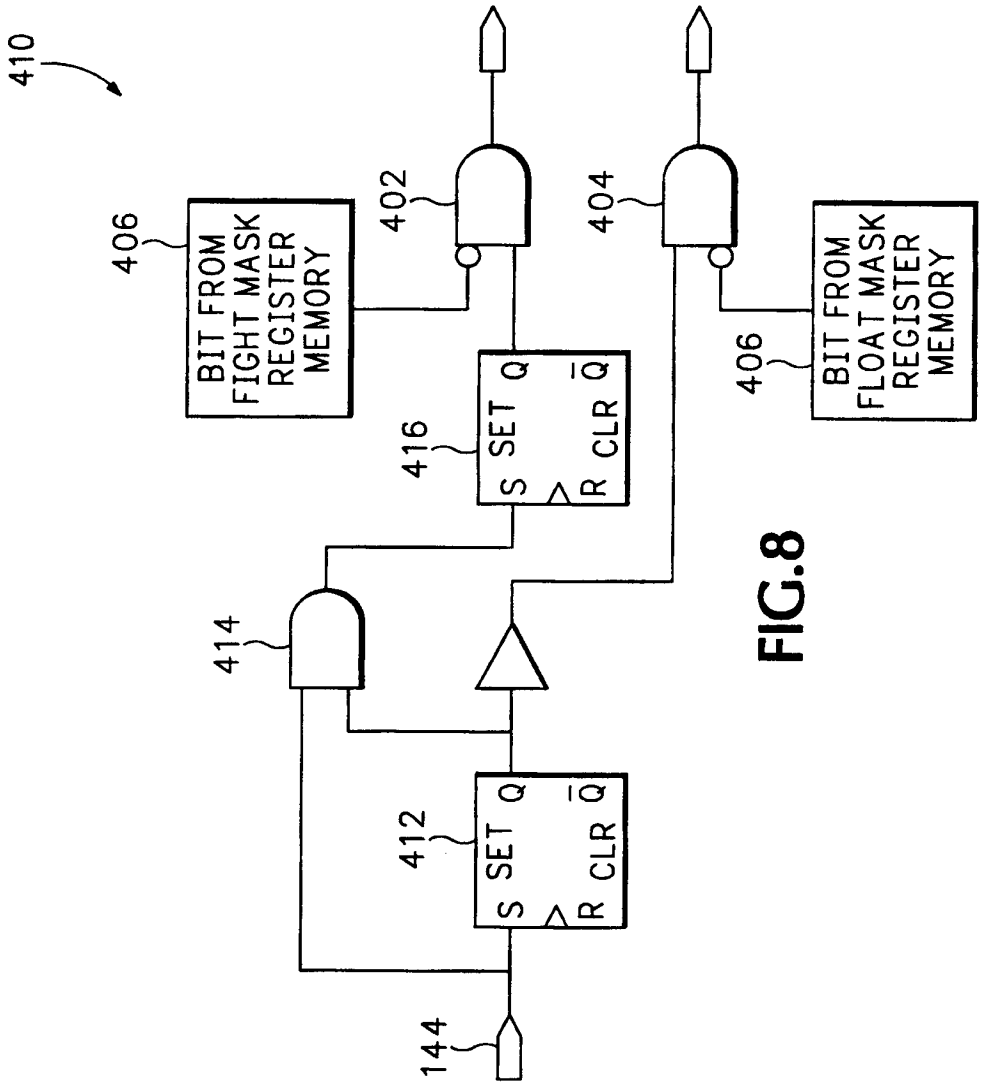

410
406
BIT FROM FIGHT MASK REGISTER MEMORY
402
404
406
BIT FROM FLOAT MASK REGISTER MEMORY
416
S SET Q
R CLR Q̄
414
412
S SET Q
R CLR Q̄
144

FIG.8

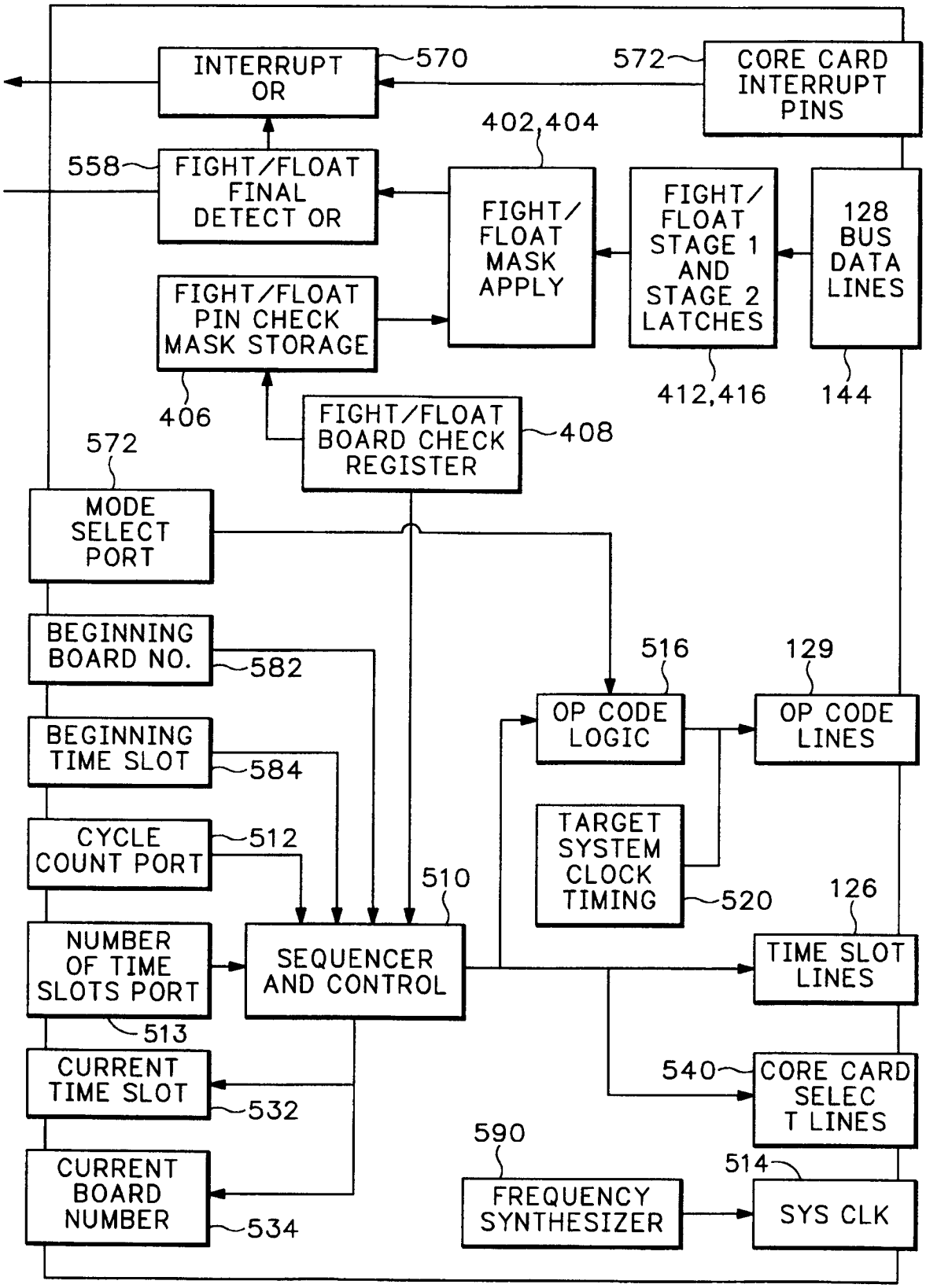

INTERRUPT OR
570
572
CORE CARD INTERRUPT PINS
402,404
558
FIGHT/FLOAT FINAL DETECT OR
FIGHT/ FLOAT MASK APPLY
FIGHT/ FLOAT STAGE 1 AND STAGE 2 LATCHES
128 BUS DATA LINES
FIGHT/FLOAT PIN CHECK MASK STORAGE
406
412,416
144
FIGHT/FLOAT BOARD CHECK REGISTER
408
572
MODE SELECT PORT
BEGINNING BOARD NO.
582
516
129
OP CODE LOGIC
OP CODE LINES
BEGINNING TIME SLOT
584
CYCLE COUNT PORT
512
TARGET SYSTEM CLOCK TIMING
520
510
126
NUMBER OF TIME SLOTS PORT
513
SEQUENCER AND CONTROL
TIME SLOT LINES
CURRENT TIME SLOT
532
540
CORE CARD SELEC T LINES
590
514
CURRENT BOARD NUMBER
534
FREQUENCY SYNTHESIZER
SYS CLK

FIG.9

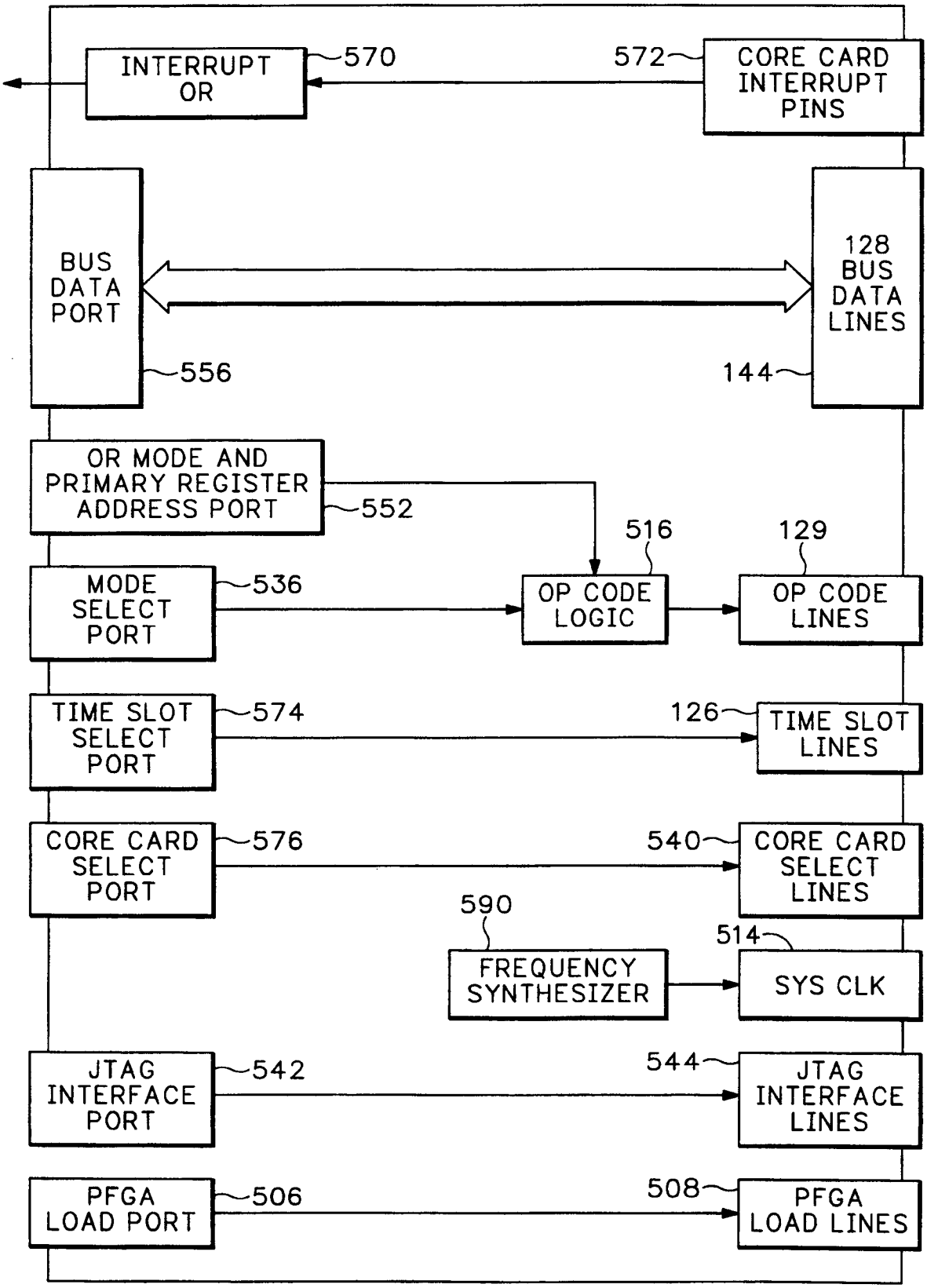

INTERRUPT OR
570
572
CORE CARD INTERRUPT PINS
BUS DATA PORT
556
128 BUS DATA LINES
144
OR MODE AND PRIMARY REGISTER ADDRESS PORT
552
516
129
MODE SELECT PORT
536
OP CODE LOGIC
OP CODE LINES
TIME SLOT SELECT PORT
574
126
TIME SLOT LINES
CORE CARD SELECT PORT
576
540
CORE CARD SELECT LINES
590
514
FREQUENCY SYNTHESIZER
SYS CLK
JTAG INTERFACE PORT
542
544
JTAG INTERFACE LINES
PFGA LOAD PORT
506
508
PFGA LOAD LINES

FIG.10

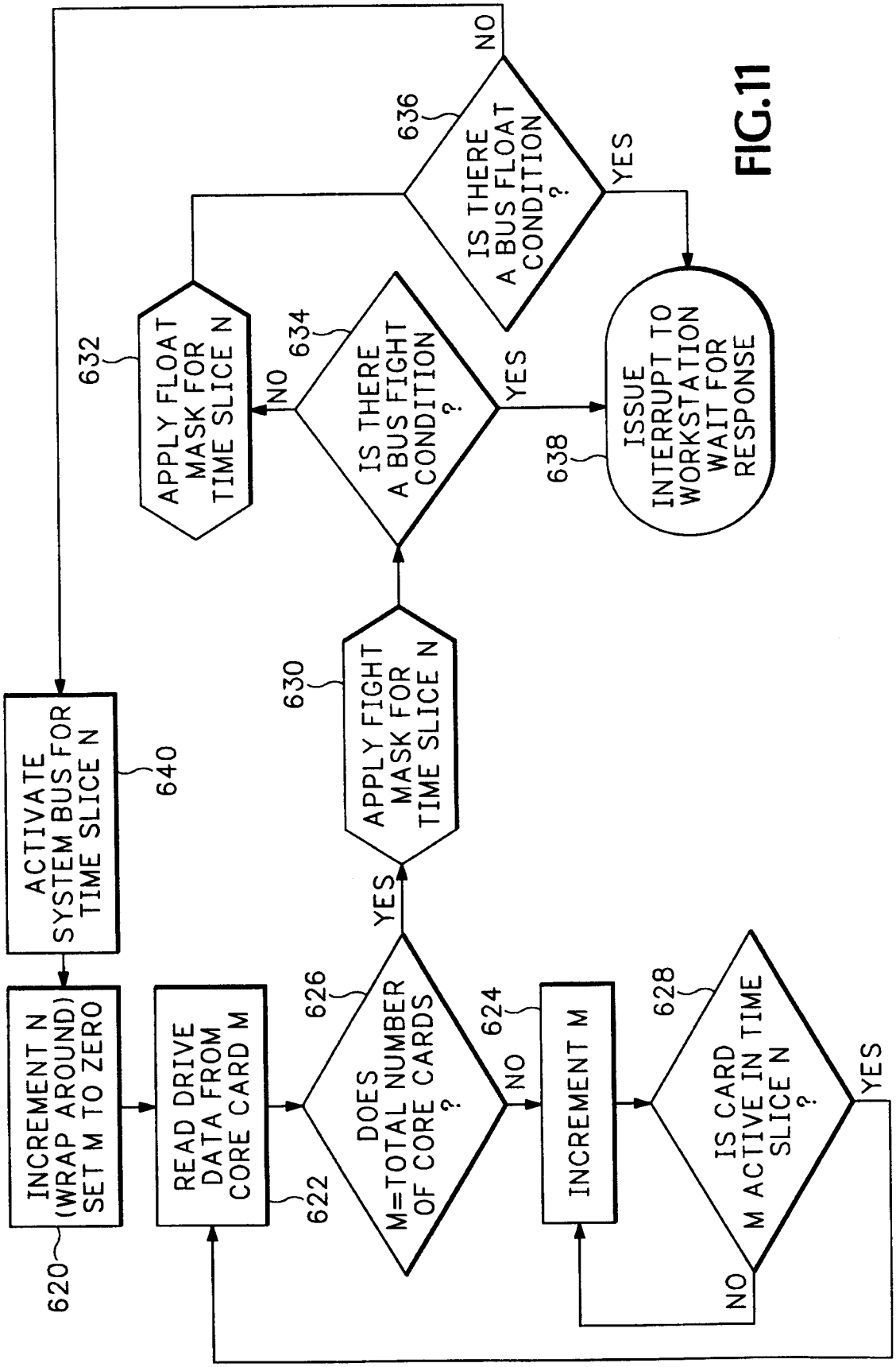

INCREMENT N (WRAP AROUND) SET M TO ZERO
620
READ DRIVE DATA FROM CORE CARD M
622
DOES M=TOTAL NUMBER OF CORE CARDS ?
626
YES
NO
INCREMENT M
624
IS CARD M ACTIVE IN TIME SLICE N ?
628
YES
NO
ACTIVATE SYSTEM BUS FOR TIME SLICE N
640
APPLY FIGHT MASK FOR TIME SLICE N
630
IS THERE A BUS FIGHT CONDITION ?
634
YES
NO
APPLY FLOAT MASK FOR TIME SLICE N
632
ISSUE INTERRUPT TO WORKSTATION WAIT FOR RESPONSE
638
IS THERE A BUS FLOAT CONDITION ?
636
YES
NO

FIG.11

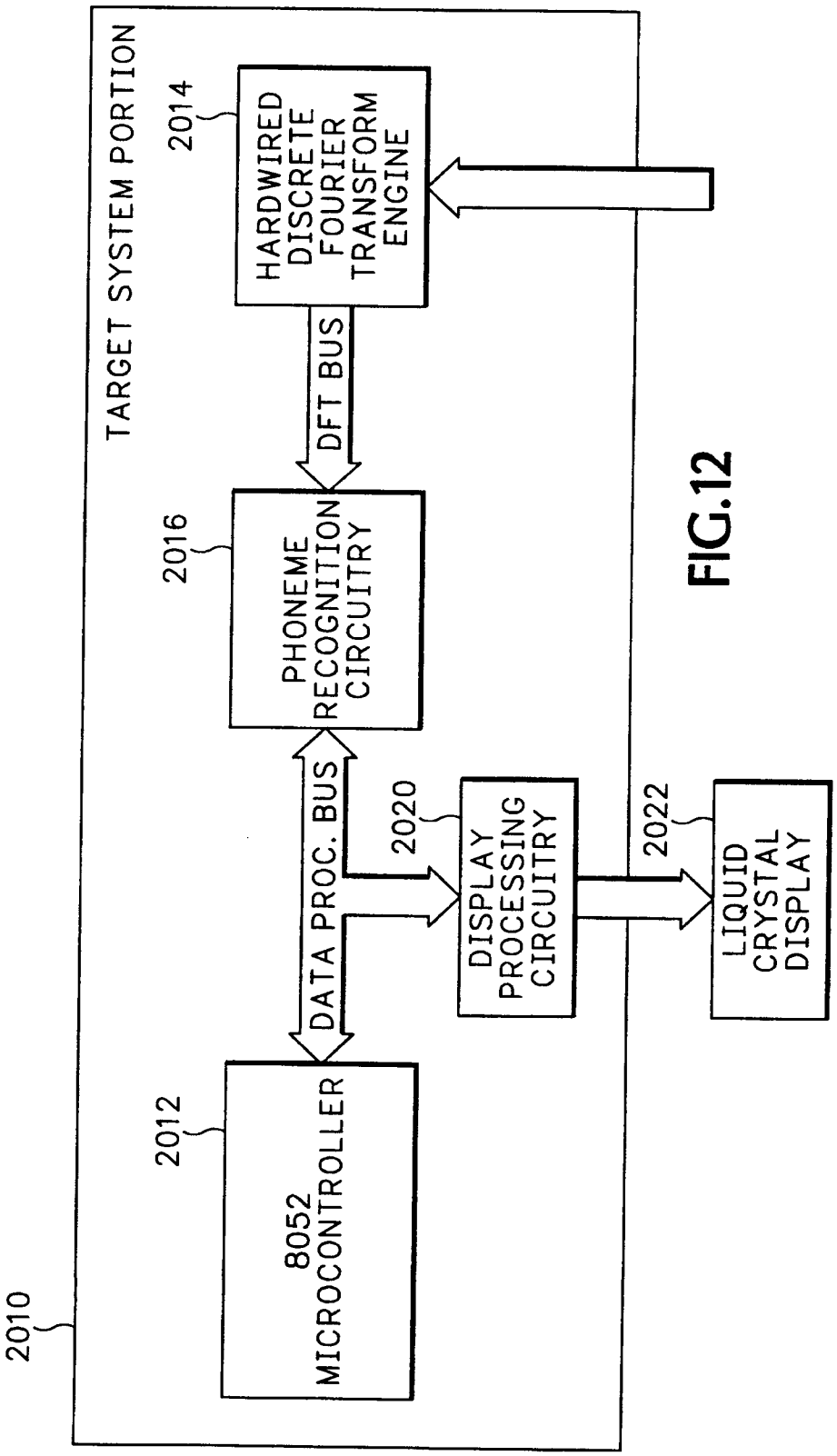

TARGET SYSTEM PORTION
2010
2012
8052 MICROCONTROLLER
2014
HARDWIRED DISCRETE FOURIER TRANSFORM ENGINE
DFT BUS
2016
PHONEME RECOGNITION CIRCUITRY
DATA PROC. BUS
2020
DISPLAY PROCESSING CIRCUITRY
2022
LIQUID CRYSTAL DISPLAY

FIG.12

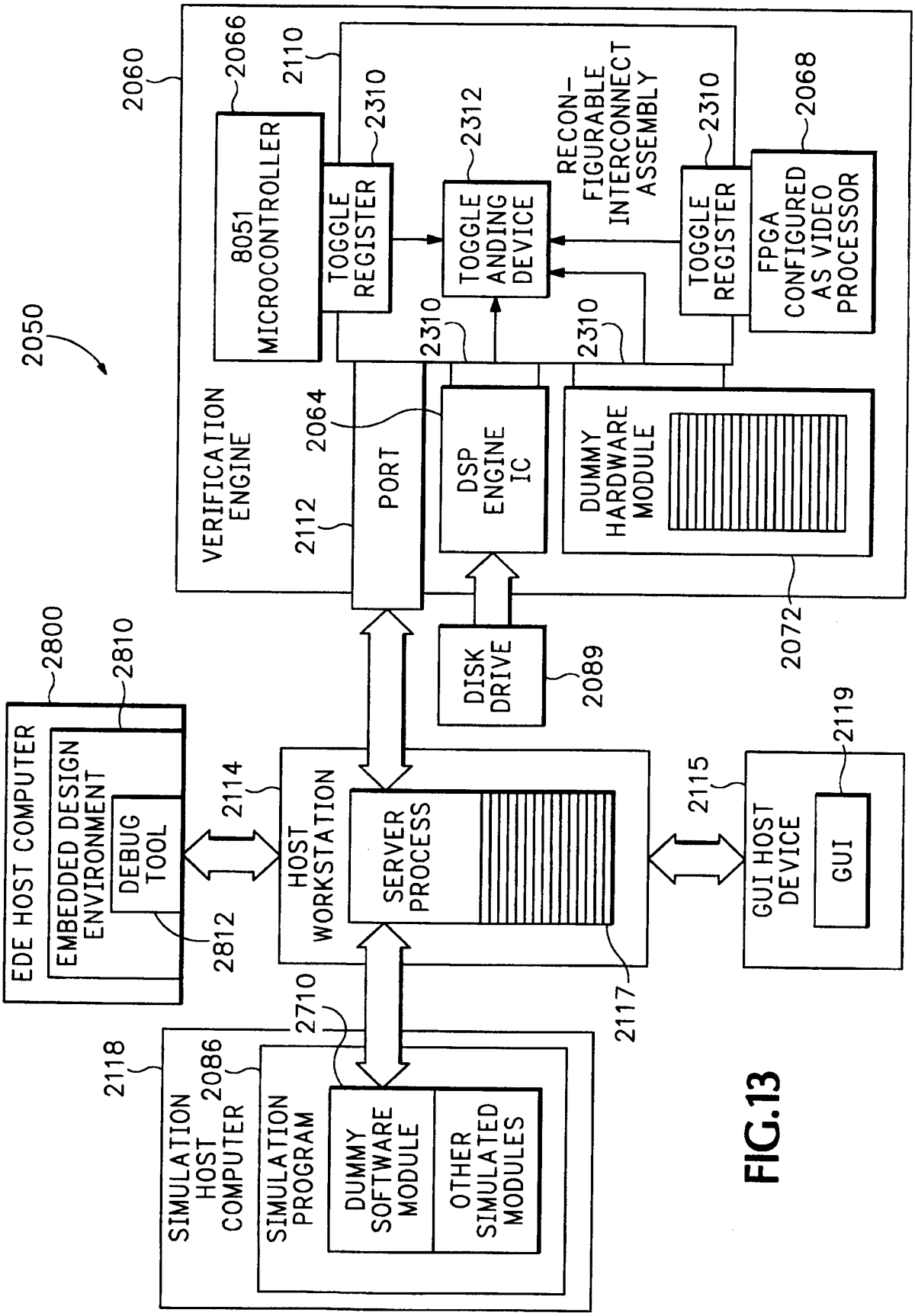

2050
2060
VERIFICATION ENGINE
2066
8051 MICROCONTROLLER
2110
2310
TOGGLE REGISTER
2312
TOGGLE ANDING DEVICE
RECON-FIGURABLE INTERCONNECT ASSEMBLY
2310
TOGGLE REGISTER
2068
FPGA CONFIGURED AS VIDEO PROCESSOR
2310
2310
2064
PORT
2112
DSP ENGINE IC
DUMMY HARDWARE MODULE
2072
2800
EDE HOST COMPUTER
2810
EMBEDDED DESIGN ENVIRONMENT
DEBUG TOOL
2812
2114
HOST WORKSTATION
SERVER PROCESS
2117
DISK DRIVE
2089
2115
GUI HOST DEVICE
2119
GUI
2710
2118
SIMULATION HOST COMPUTER
2086
SIMULATION PROGRAM
DUMMY SOFTWARE MODULE
OTHER SIMULATED MODULES

FIG.13

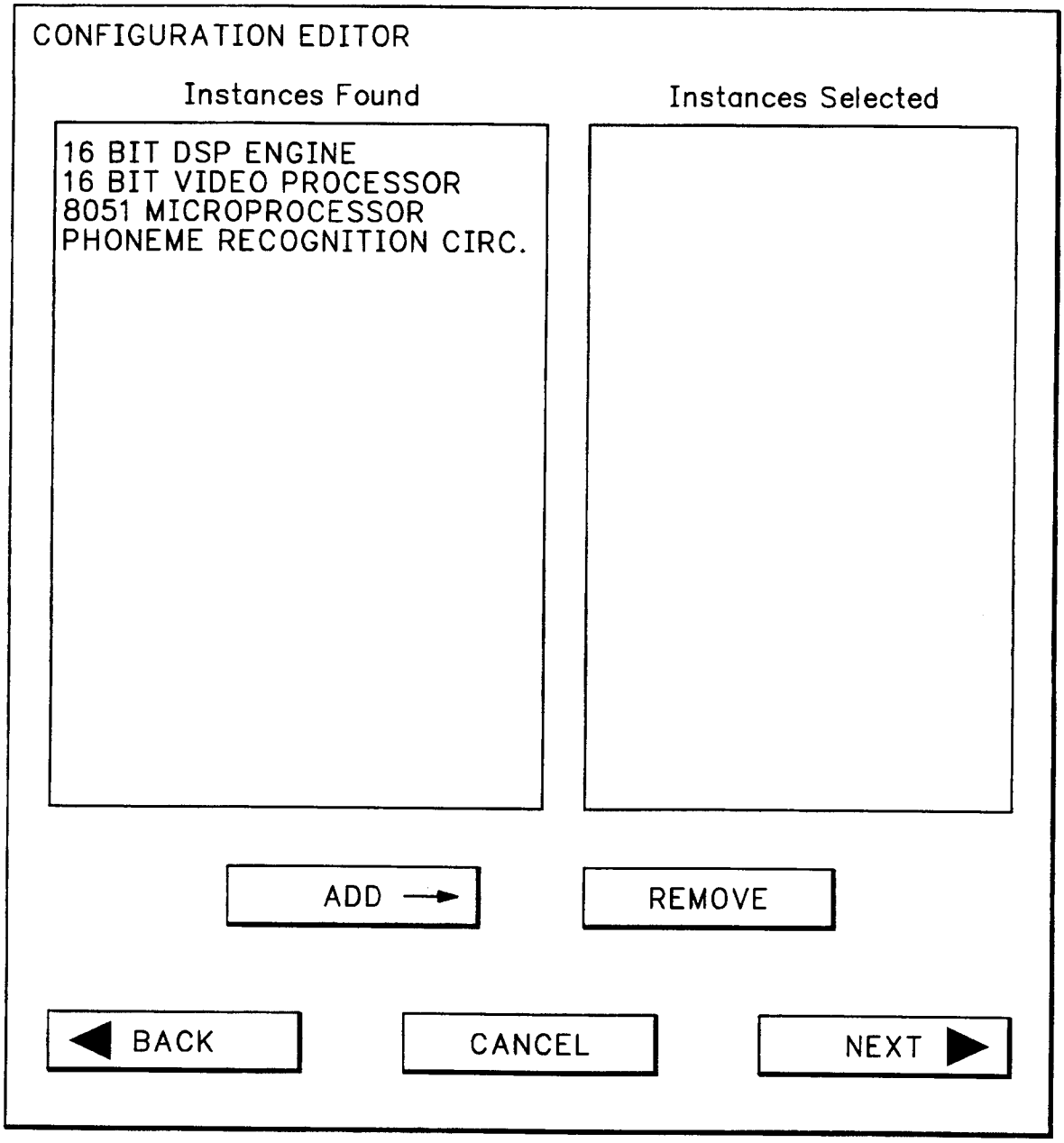

CONFIGURATION EDITOR
Instances Found
Instances Selected
16 BIT DSP ENGINE
16 BIT VIDEO PROCESSOR
8051 MICROPROCESSOR
PHONEME RECOGNITION CIRC.
ADD
REMOVE
BACK
CANCEL
NEXT

FIG.14

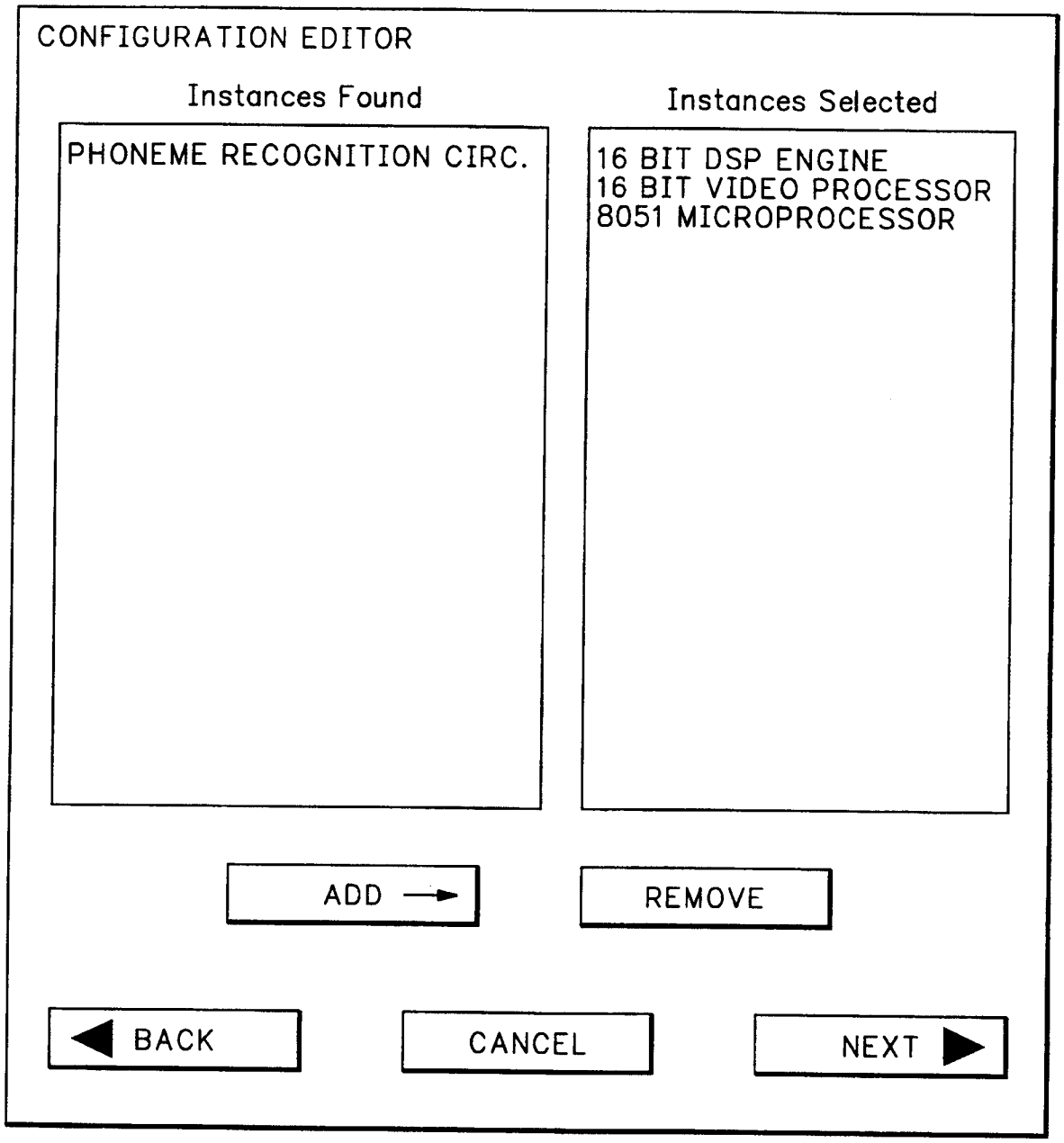

CONFIGURATION EDITOR
Instances Found
Instances Selected
PHONEME RECOGNITION CIRC.
16 BIT DSP ENGINE
16 BIT VIDEO PROCESSOR
8051 MICROPROCESSOR
ADD
REMOVE
BACK
CANCEL
NEXT

FIG.15

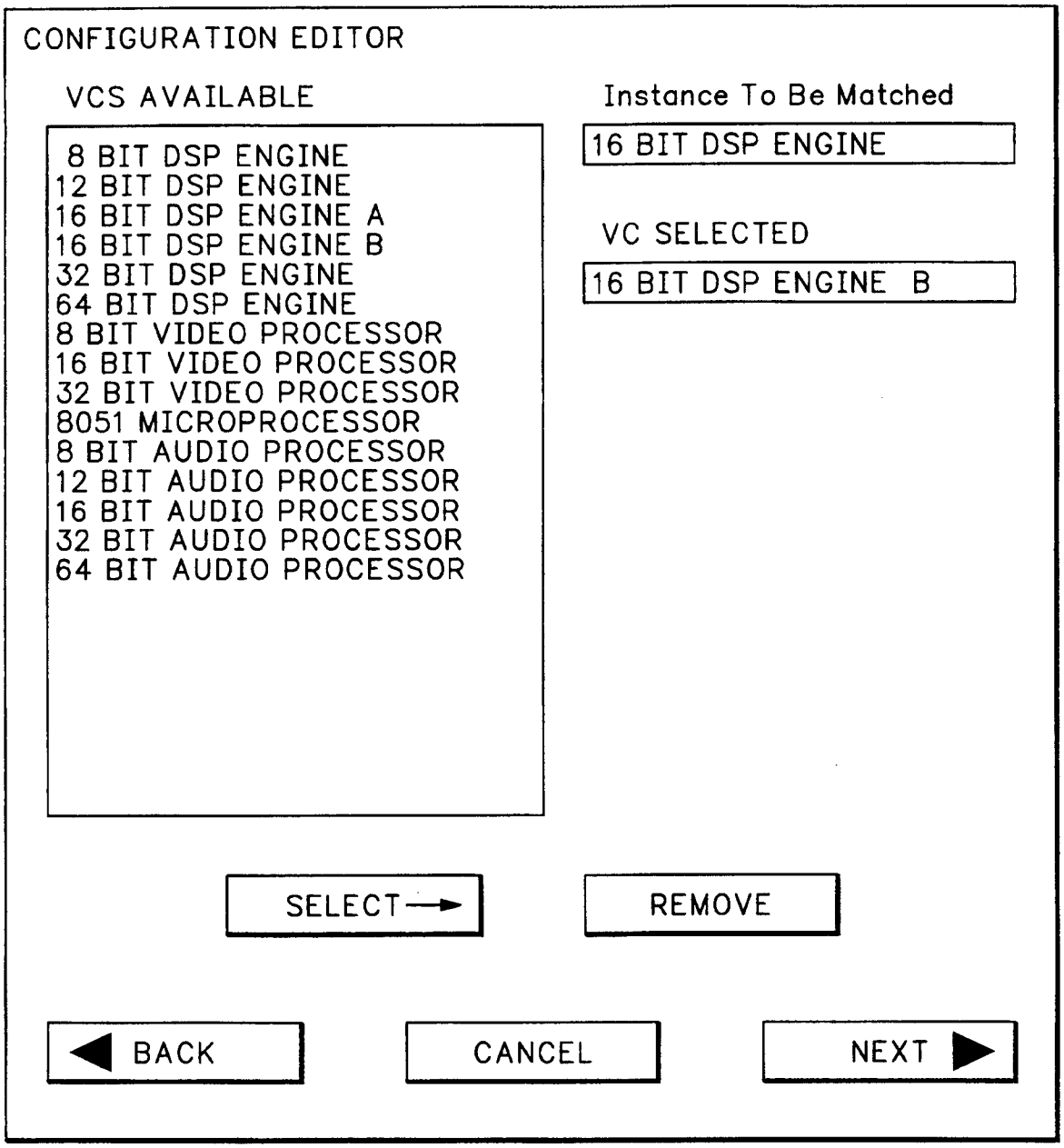

CONFIGURATION EDITOR
VCS AVAILABLE
8 BIT DSP ENGINE
12 BIT DSP ENGINE
16 BIT DSP ENGINE A
16 BIT DSP ENGINE B
32 BIT DSP ENGINE
64 BIT DSP ENGINE
8 BIT VIDEO PROCESSOR
16 BIT VIDEO PROCESSOR
32 BIT VIDEO PROCESSOR
8051 MICROPROCESSOR
8 BIT AUDIO PROCESSOR
12 BIT AUDIO PROCESSOR
16 BIT AUDIO PROCESSOR
32 BIT AUDIO PROCESSOR
64 BIT AUDIO PROCESSOR
Instance To Be Matched
16 BIT DSP ENGINE
VC SELECTED
16 BIT DSP ENGINE B
SELECT
REMOVE
BACK
CANCEL
NEXT

FIG.16

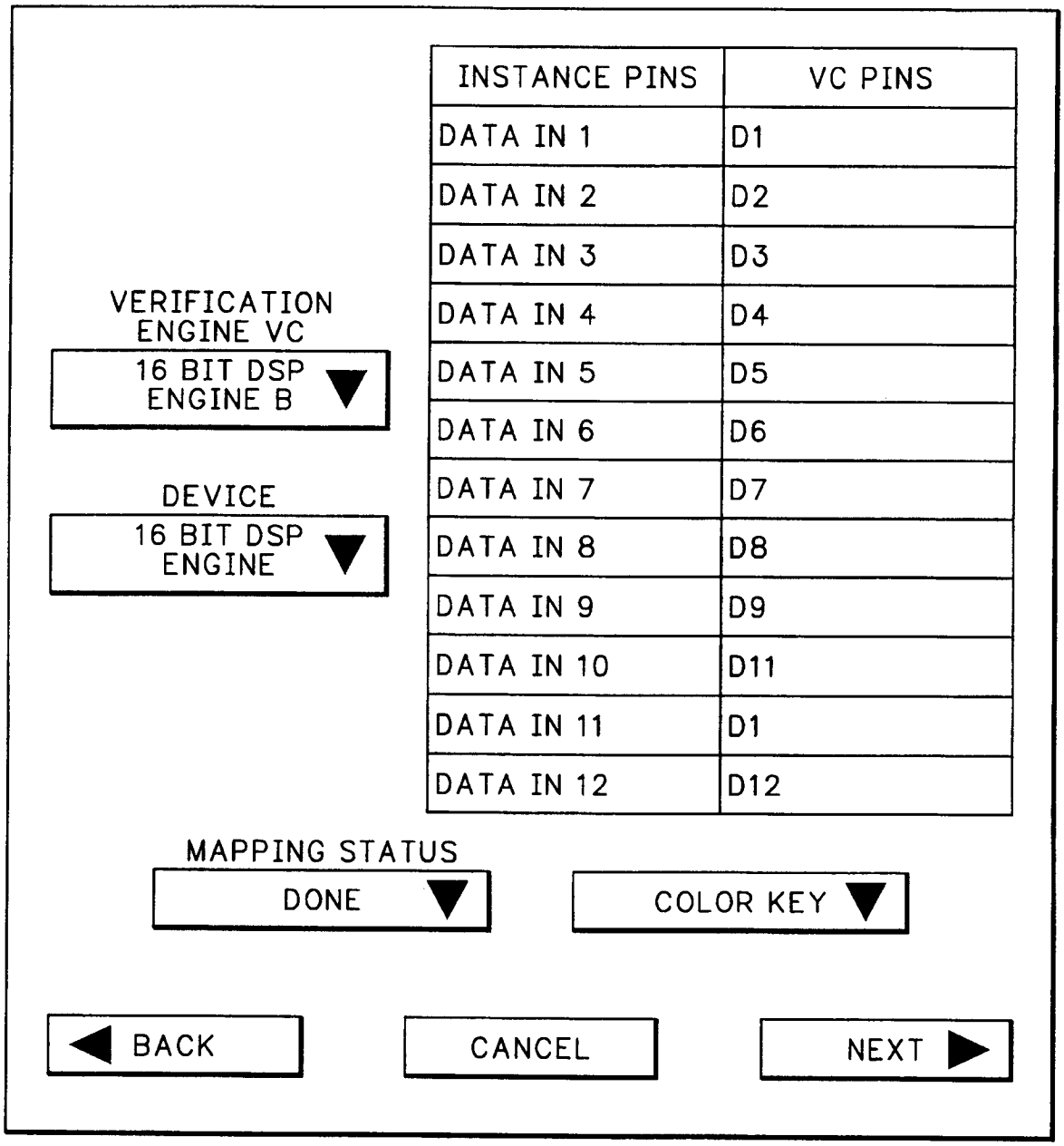

INSTANCE PINS
VC PINS
DATA IN 1
D1
DATA IN 2
D2
DATA IN 3
D3
DATA IN 4
D4
DATA IN 5
D5
DATA IN 6
D6
DATA IN 7
D7
DATA IN 8
D8
DATA IN 9
D9
DATA IN 10
D11
DATA IN 11
D1
DATA IN 12
D12
VERIFICATION ENGINE VC
16 BIT DSP ENGINE B
DEVICE
16 BIT DSP ENGINE
MAPPING STATUS
DONE
COLOR KEY
BACK
CANCEL
NEXT

FIG.17

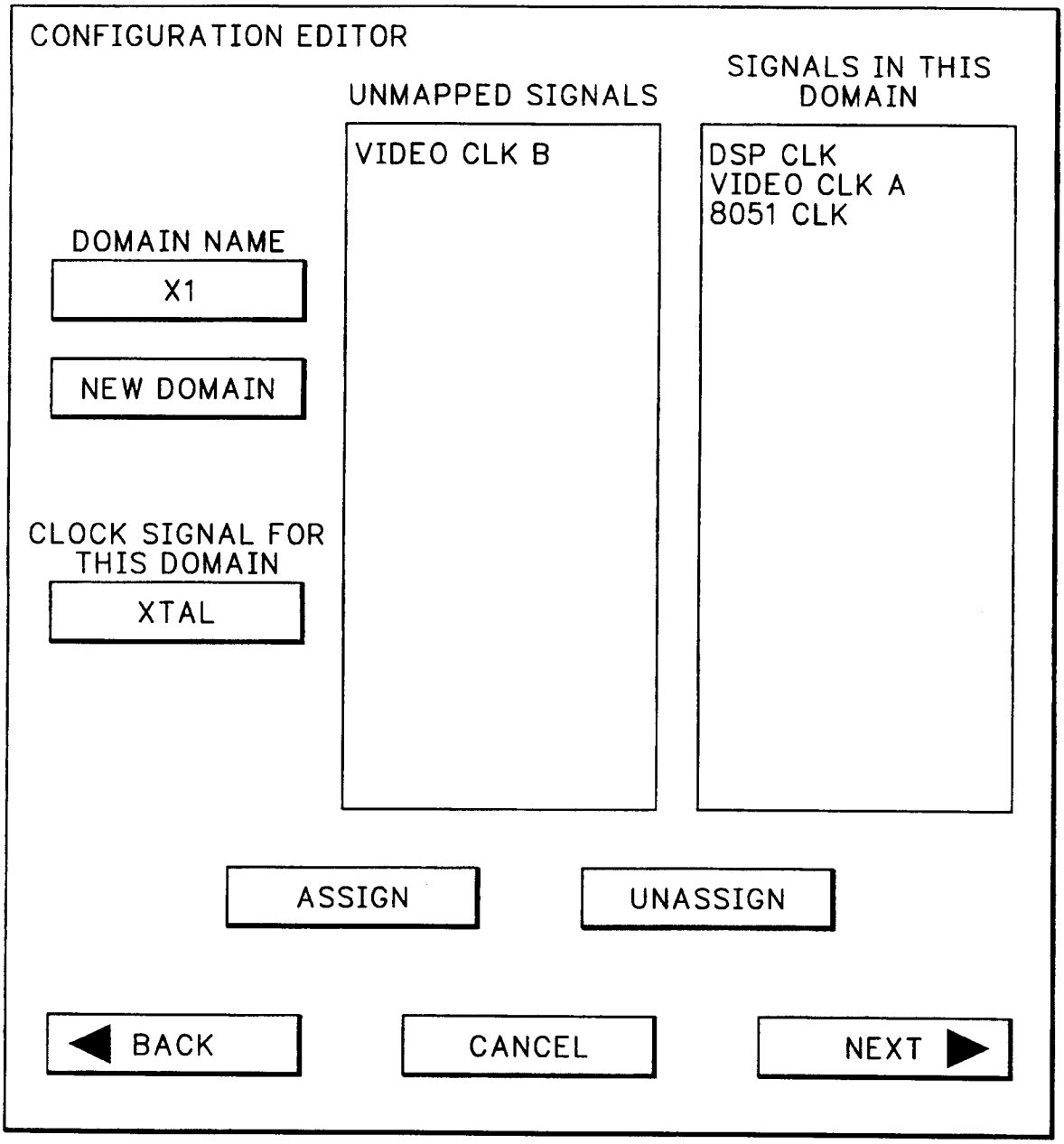

CONFIGURATION EDITOR
UNMAPPED SIGNALS
SIGNALS IN THIS DOMAIN
VIDEO CLK B
DSP CLK
VIDEO CLK A
8051 CLK
DOMAIN NAME
X1
NEW DOMAIN
CLOCK SIGNAL FOR THIS DOMAIN
XTAL
ASSIGN
UNASSIGN
BACK
CANCEL
NEXT

FIG.18

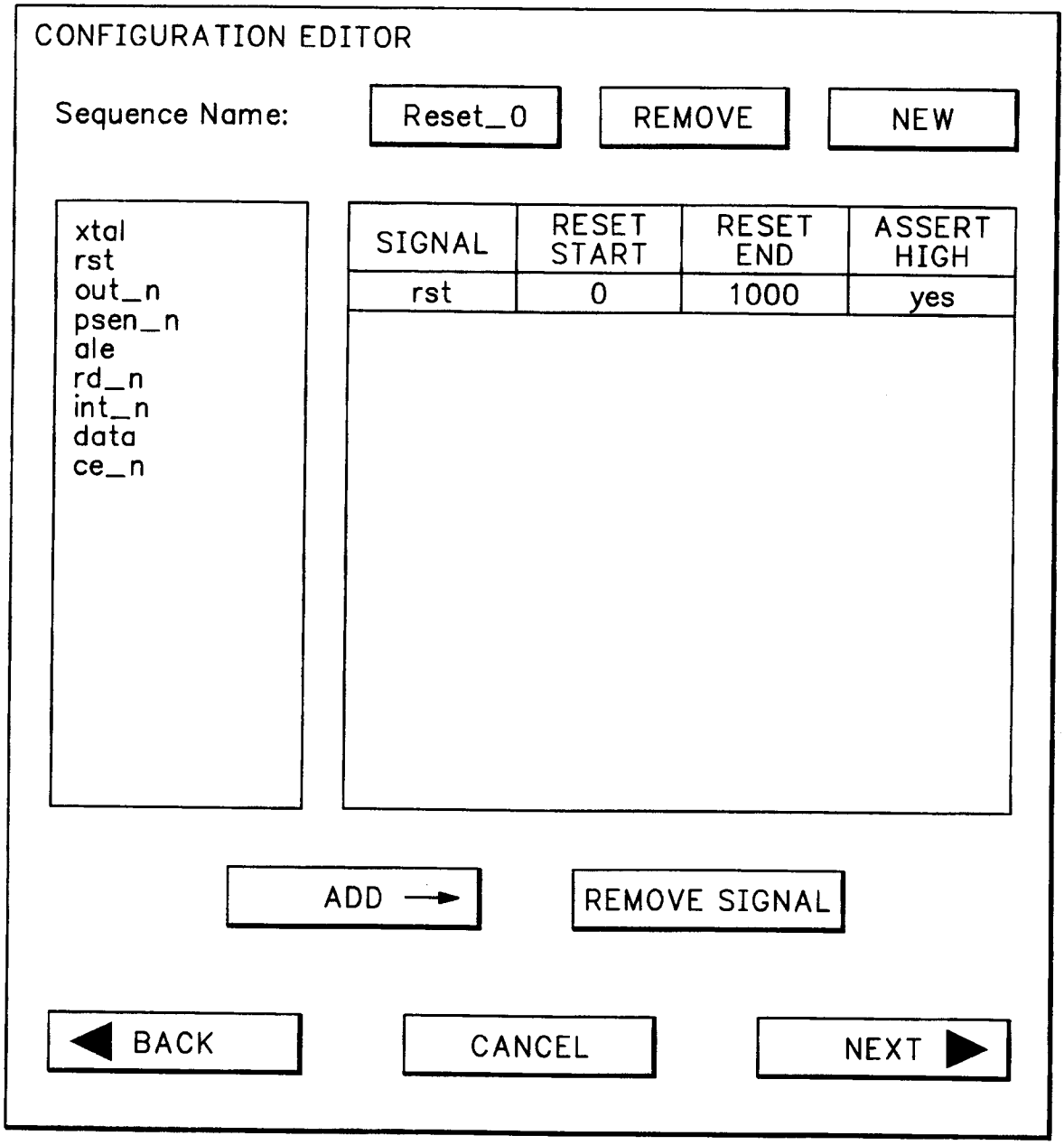

CONFIGURATION EDITOR
Sequence Name:
Reset_0
REMOVE
NEW
xtal
rst
out_n
psen_n
ale
rd_n
int_n
data
ce_n
SIGNAL
RESET START
RESET END
ASSERT HIGH
rst
0
1000
yes
ADD
REMOVE SIGNAL
BACK
CANCEL
NEXT

FIG.19

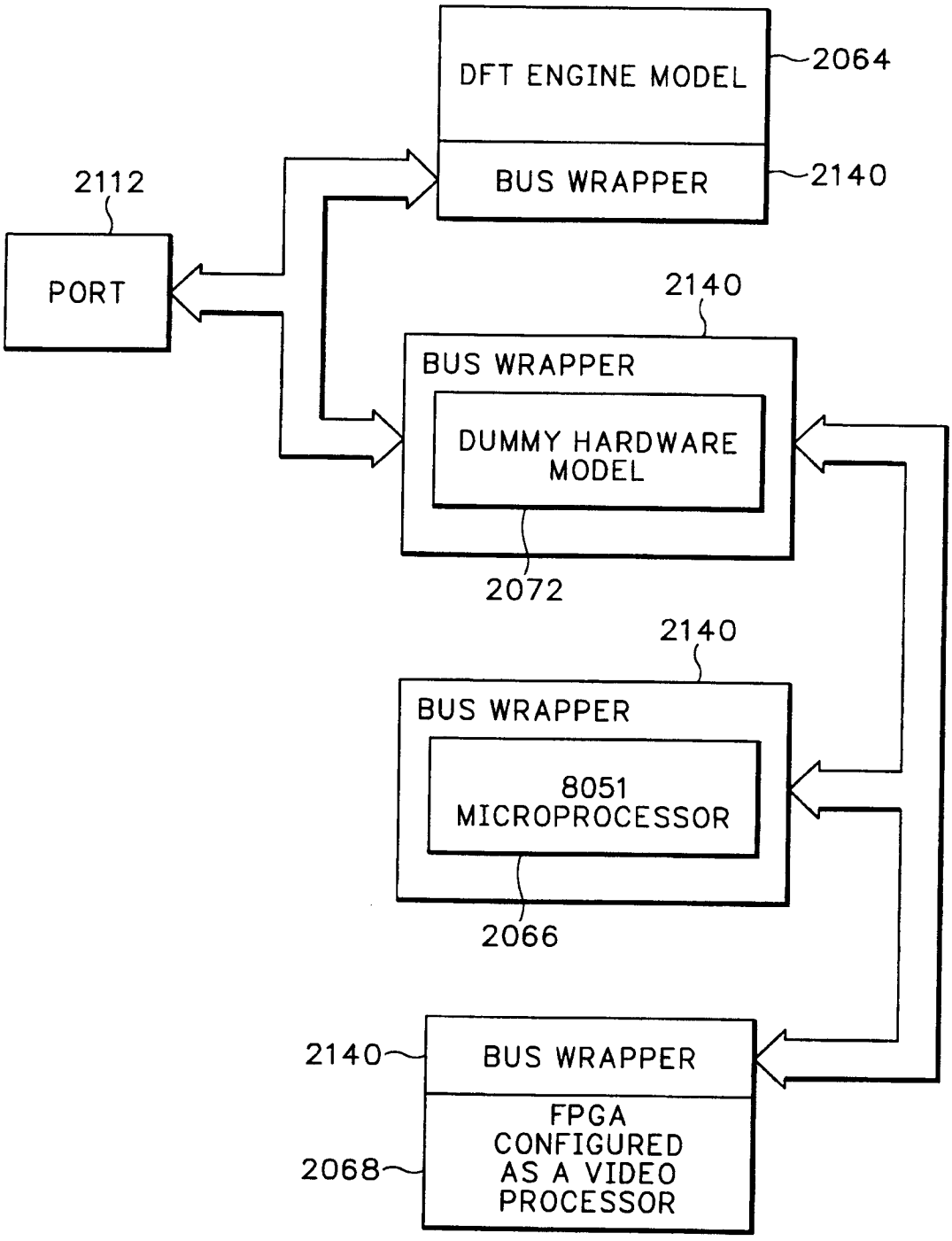

DFT ENGINE MODEL
2064
BUS WRAPPER
2140
2112
PORT
2140
BUS WRAPPER
DUMMY HARDWARE MODEL
2072
2140
BUS WRAPPER
8051 MICROPROCESSOR
2066
2140
BUS WRAPPER
FPGA CONFIGURED AS A VIDEO PROCESSOR
2068

FIG.20

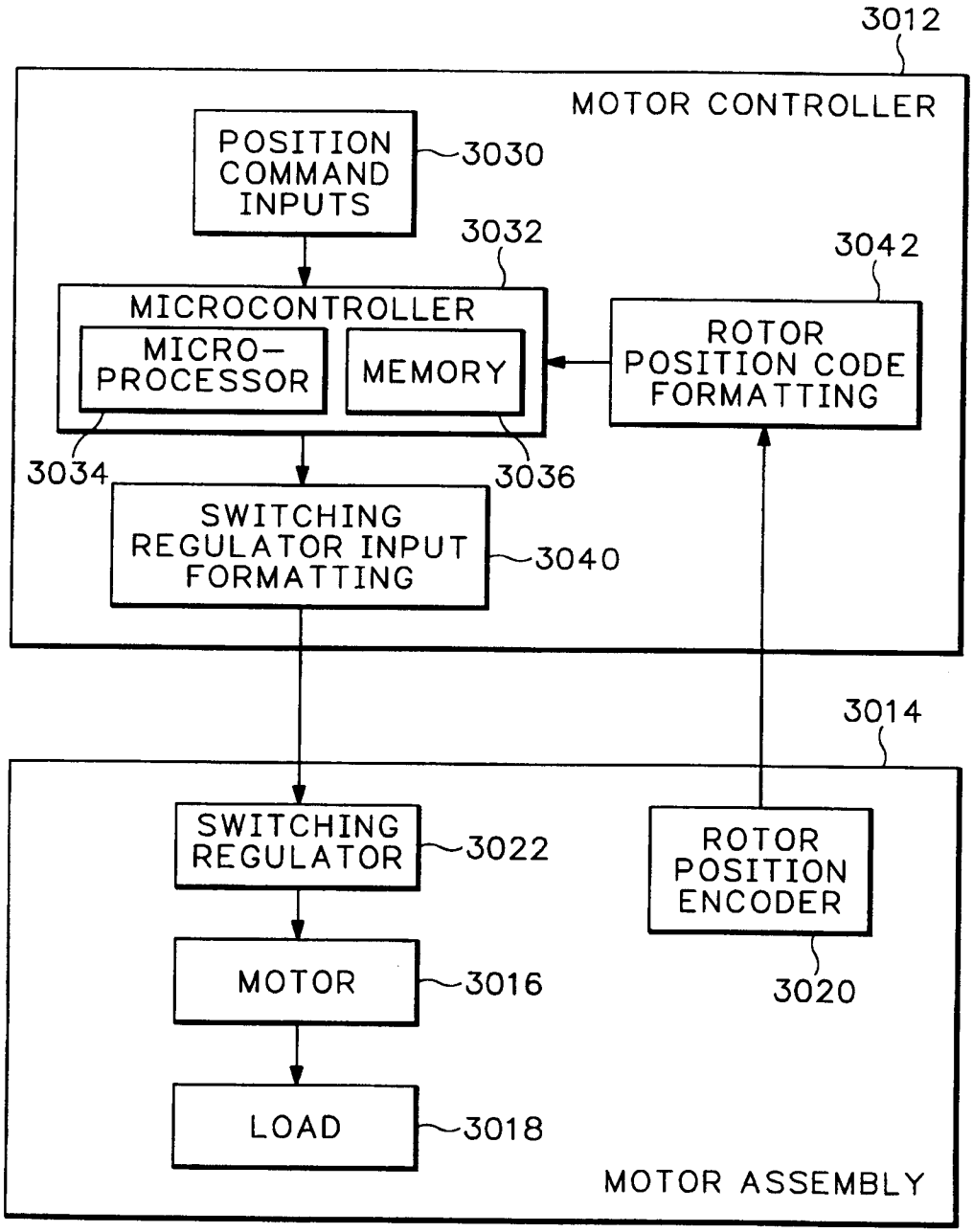

3012
MOTOR CONTROLLER
POSITION COMMAND INPUTS
3030
3032
3042
MICROCONTROLLER
MICRO-PROCESSOR
MEMORY
ROTOR POSITION CODE FORMATTING
3034
3036
SWITCHING REGULATOR INPUT FORMATTING
3040
3014
SWITCHING REGULATOR
3022
ROTOR POSITION ENCODER
MOTOR
3016
3020
LOAD
3018
MOTOR ASSEMBLY

FIG.21

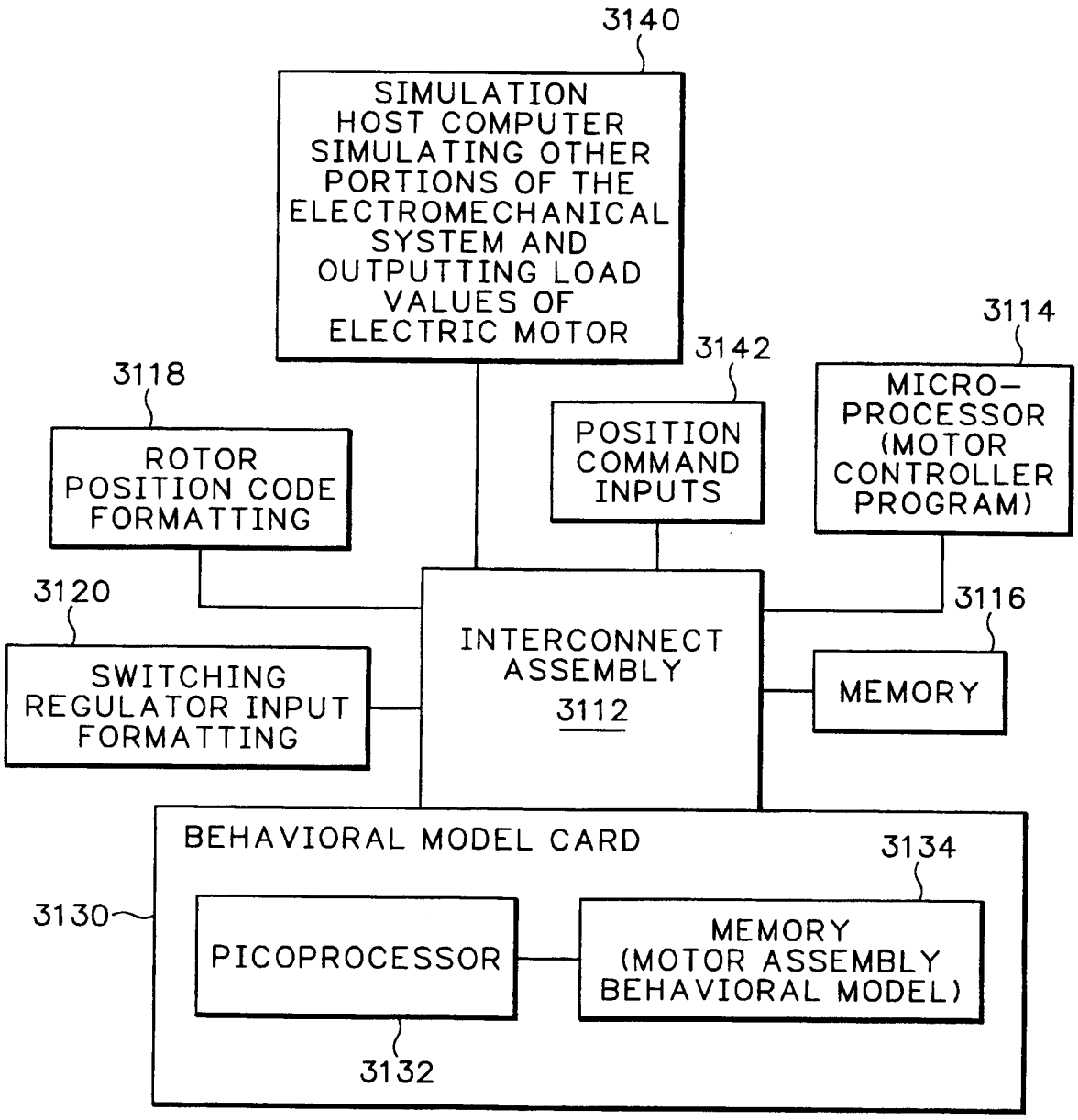

3140
SIMULATION HOST COMPUTER SIMULATING OTHER PORTIONS OF THE ELECTROMECHANICAL SYSTEM AND OUTPUTTING LOAD VALUES OF ELECTRIC MOTOR
3114
3142
3118
MICRO-PROCESSOR (MOTOR CONTROLLER PROGRAM)
POSITION COMMAND INPUTS
ROTOR POSITION CODE FORMATTING
3120
3116
INTERCONNECT ASSEMBLY 3112
SWITCHING REGULATOR INPUT FORMATTING
MEMORY
BEHAVIORAL MODEL CARD
3134
3130
PICOPROCESSOR
MEMORY (MOTOR ASSEMBLY BEHAVIORAL MODEL)
3132

FIG.22

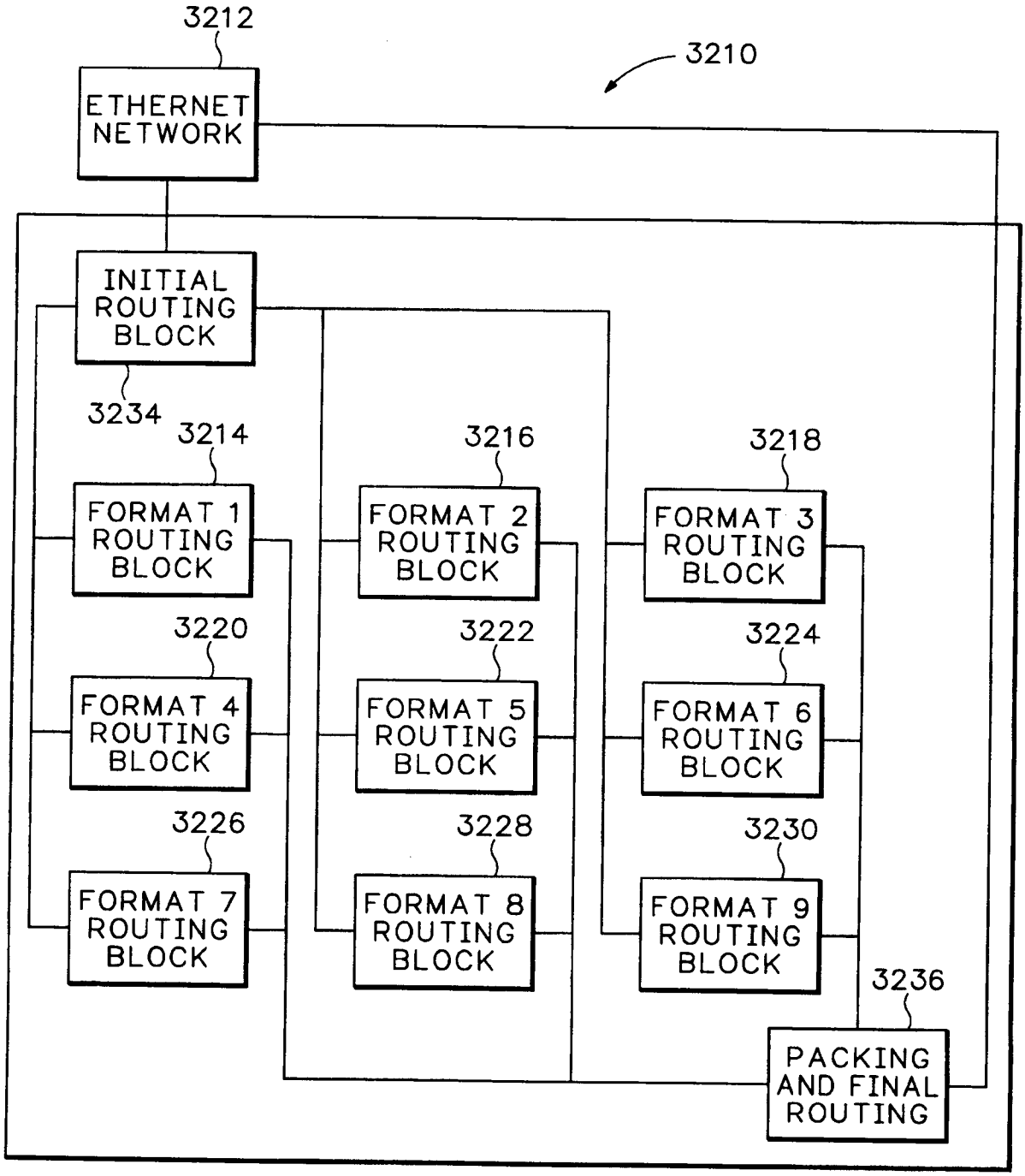

3212
3210
ETHERNET NETWORK
INITIAL ROUTING BLOCK
3234
3214
3216
3218
FORMAT 1 ROUTING BLOCK
FORMAT 2 ROUTING BLOCK
FORMAT 3 ROUTING BLOCK
3220
3222
3224
FORMAT 4 ROUTING BLOCK
FORMAT 5 ROUTING BLOCK
FORMAT 6 ROUTING BLOCK
3226
3228
3230
FORMAT 7 ROUTING BLOCK
FORMAT 8 ROUTING BLOCK
FORMAT 9 ROUTING BLOCK
3236
PACKING AND FINAL ROUTING

FIG.23

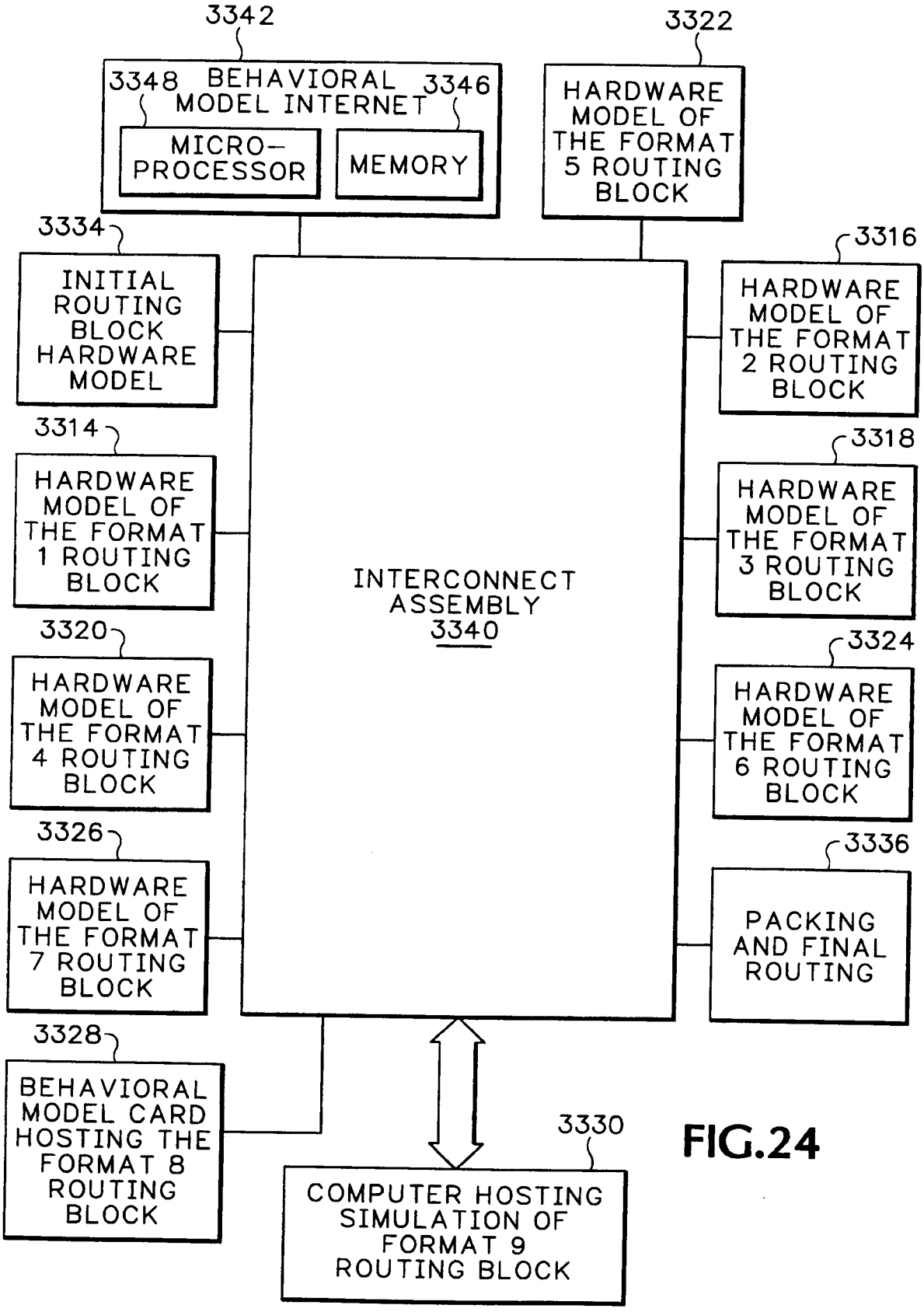

3342
3348 BEHAVIORAL MODEL INTERNET 3346
MICRO-PROCESSOR
MEMORY
3322
HARDWARE MODEL OF THE FORMAT 5 ROUTING BLOCK
3334
INITIAL ROUTING BLOCK HARDWARE MODEL
3316
HARDWARE MODEL OF THE FORMAT 2 ROUTING BLOCK
3314
HARDWARE MODEL OF THE FORMAT 1 ROUTING BLOCK
3318
HARDWARE MODEL OF THE FORMAT 3 ROUTING BLOCK
INTERCONNECT ASSEMBLY 3340
3320
HARDWARE MODEL OF THE FORMAT 4 ROUTING BLOCK
3324
HARDWARE MODEL OF THE FORMAT 6 ROUTING BLOCK
3326
HARDWARE MODEL OF THE FORMAT 7 ROUTING BLOCK
3336
PACKING AND FINAL ROUTING
3328
BEHAVIORAL MODEL CARD HOSTING THE FORMAT 8 ROUTING BLOCK
3330
COMPUTER HOSTING SIMULATION OF FORMAT 9 ROUTING BLOCK

FIG.24

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/00292

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06F 17/50, 13/10, 13/12, 9/455

US CL : 716/4, 5, 16, 18; 703/20, 25

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 716/4, 5, 16, 18; 703/20, 25

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

Please See Extra Sheet.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X --- Y	US 5,794,012 A (AVERILL) 11 August 1998; Fig. 3; col. 6, line 60 to col. 9, line 32, wherein the depiper provides for the necessary control blocks functions to the synchronizer (i.e., system controller) and also functions as the multiplexer; col. 5, line 39 to col. 6, line 29, wherein the I/O adapters provide for the toggle coverage detection.	1-10,22-24, 27-22, 31-33, 41-46, 48- 50; 34 --- 30,47

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 16 MAY 2000	Date of mailing of the international search report 13 JUN 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer Paul R. Lintz Telephone No. (703) 305-3832

Form PCT/ISA/210 (second sheet) (July 1998) *

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/00292

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,625,580 A (READ et al) 29 April 1997, col. 4, line 2 to col. col. 9, line 51, wherein the PEL provides for the hus wrapper/controller, drivers, and the TG provides for the sytem controller for timing synchronization;	1-10,22-24, 27-34, 41-50;
	the driver determining direction is described in col. 53, line 35 to col. 74;	11-21, 25-26;
	the user interactive system is described in col. 1, lines 9-15 and col. 13, line 38 to col. 22, line 10.	35-40
Y,P	US 6,006,022 A (RHIM et al) 21 December 1999, Fig. 1, col. 6, lines 28-67.	30, 47
Y	US 4,744,084 A (BECK et al) 10 May 1988, Fig. 1, col. 21, line 19 to col. 29.	30, 47
A, P	US 5,991,529 A (COX et al) 23 November 1999, Fig. 3, col. 2, lines 20-51.	1-50
A	US 5,483,640 A (ISFELD et al) 09 January 1996, Fig. 1, col. 1, line 50 to col. 3, line 21.	1-50
A	US 5,596,742 A (AGARWAL et al) 21 January 1997, Fig. 3, col. 1, line 55 to col. 3, line 35.	1-50
A	US 5,649,176 A (SELVIDGE et al) 15 July 1997, Fig. 4A, col. 2, line 43 to col. 4, line 20.	1-50
A	US 4,357,678 A (DAVIS) 02 November 1982, abstract, Fig. 7.	1-50
A	US 5,479,355 A (HYDUKE) 26 December 1995, abstract, Fig. 1.	1-50

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/00292

B. FIELDS SEARCHED
Electronic data bases consulted (Name of data base and where practicable terms used):

BRS (USPATFULL, EPO, JPO), IEE/IEEE
Search terms: time, multiplex$4, bus, interconnect$5, line, connect$5, hardware, model$4, input/output, i/o, input, output, switch$5, synchroni$